Opportunistic Machine Learning Methods for Effective Insider Threat Detection

Diana Haidar

PhD in Computing

2018

Birmingham City University

BIRMINGHAM CITY UNIVERSITY

DOCTORAL THESIS

Opportunistic Machine Learning Methods for Effective Insider Threat Detection

Author:
Diana HAIDAR

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Enterprise Systems & Centre for Cyber Security
School of Computing and Digital Technology

November 2018

Declaration of Authorship

I, Diana HAIDAR, declare that this thesis titled, “Opportunistic Machine Learning Methods for Effective Insider Threat Detection” and the work presented in it are my own. I confirm that:

- This work was done wholly while in candidature for a research degree at this University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Never doubt that a small group of thoughtful, committed citizens can change the world; indeed, it's the only thing that ever has."

Margaret Mead

Abstract

The topic of insider threat detection is getting an increased concern from academia, industry, and governments due to the growing number of malicious insider incidents. A malicious insider threat is devised of a set of anomalous behaviours attributed to an insider who exploit their privileges with the intention to compromise the confidentiality, integrity, or availability of the system or data.

The existing approaches for detecting insider threats still have a common shortcoming, which is the high number of false alarms (false positives), which deceives the system administrator(s) about suspicious behaviour of many users. To address the shortcoming of false alarms, in this thesis, we formulate an *opportunistic approach* to detect insider threats with the aim of any-behaviour-all-threat detection. As a preliminary step, we apply feature engineering on the data logs of users' behaviour. This work is conducted on synthetic CMU-CERT data sets which implement a variety of malicious insider threat scenarios.

The maturity of data in an organisation is defined into three cases based on the availability of labelled data. We address the different cases of data maturity by proposing, developing, and evaluating machine learning approaches that incorporate techniques to reduce false alarms. The first presents a class imbalance approach, namely CD-AMOTRE, which combines the concept of Class Decomposition (CD) and a novel Artificial Minority Oversampling and Trapper REmoval (AMOTRE) technique. The second builds an adaptive one-class ensemble-based anomaly detection framework which introduces a progressive update method with an outlier-aware artificial oversampling procedure. The third proposes a real-time anomaly detection approach, namely Ensemble of Random subspace Anomaly detectors In Data Streams (E-RAIDS). The proposed approaches detect most/all of the malicious insider threats, and achieve the minimum FP over the data sets compared to the existing machine learning approaches.

Acknowledgements

First and Foremost, all praise to Almighty GOD for giving me the blessing, and endurance to complete this PhD study.

The PhD research journey is said to be an extremely challenging process – *like climbing a mountain* – to achieve a new success in your life journey. The uncountable days the researcher invests climbing this mountain would be unbearable without the assistance, guidance, and support of various people. My sincere gratitude goes to all those who have been part of my experience. Given that I want to follow the expectations of people reading this, I will follow the tradition in the acknowledgements order. I would like to take this opportunity to acknowledge the following people who have helped me during this PhD research journey.

Prof. Mohamed Medhat GABER, *Director of Studies*, for his deep knowledge, insights, guidance, argument, and academic support over the course of my PhD journey. I have learnt a lot and my gratitude goes to him for this.

Dr. Yevgeniya KOVALCHUK, *PhD Second Supervisor*, who has joined the supervision team in the thesis writing stage, for her support, constructive feedback, and attention to detail which helped me to shape this thesis.

Prof. Ali ABDALLAH, *PhD Second Supervisor*, who was the first to encourage me to commence my PhD at Birmingham City University, for his guidance and support to find the focus of my research proposal.

Prof. Hanifa SHAH, *Associate Dean Research and Mentor*, for her assistance and support throughout the different stages of this research. I sincerely appreciate her valuable advices which helped me to cope with some of the challenges I faced through my PhD.

Prof. Peter Larkham, *Director of Research*, who I commenced the 'Postgraduate Certificate in Research Practice' course with him along the start of the PhD course, for his research training and academic guidance to develop my academic research skills. I appreciate his proofreading assistance and feedback to prepare this thesis for submission.

Mr. Ian MCDONALD, *Former Research Officer*, for his administrative assistance for this research and for organising training events such as the 'Three-Minute Thesis Competition' to develop our communication and presentation skills. I would like to extend my thanks to him as a friend for his encouragement, support, and valuable advices throughout the different stages of this research. I sincerely appreciate him joining me in my alone library days during the thesis writing up.

Ms. Sue WITTON and Ms. Bernadette ALLEN, *Doctoral Research College*, for their administrative and technical assistance for this research and for being always helpful.

Ms. Kathy HUDSON, Personal Assistant to Associate Dean, for being always kind and helpful.

I would like to give a special acknowledgement to the *PhD committee* who will invest time to assess this research and evaluate the PhD defence.

I would like to extend my acknowledgements to my *colleagues and fellow PhD students* for their help and encouragement throughout the PhD journey.

I would like to express my deep gratitude and thanks to my *parents* Mr. Hammad HAIDAR and Mrs. Hayat AOUAD for their confidence and unconditional love. Without their encouragement and support, I could not have made it through this challenging experience. It is because of them, I am who I am today.

I would like to extend my thanks to my *siblings* Mr. Firas HAIDAR, Miss. Razan HAIDAR, Miss. Zahraa HAIDAR, and Mr. Hasan HAIDAR who make happiness worth sharing and make dreams worth believing.

A special thanks goes to my *friends*, especially Miss Aaisha REHMAN, Miss Lamya OMAR, and Miss Fatima ABDALLAH, for their support and love. It was their encouragement that raised me up again when I got weary.

Contents

List of Figures

List of Tables

List of Abbreviations

ADASYN	**ADA**aptive **SYN** sampling technique
AnyOut	**Any**time **Out**lier detection
AMOTRE	**A**rtificial **M**inority **O**versampling and textbfTrapper **RE**moval
BBAC	**B**ehaviour **B**ased **A**ccess **C**ontrol
BAIT	**B**ehavioural **A**nalysis of **I**nsider **T**hreat
BIRCH	**B**alanced **I**terative **R**educing and **C**lustering using **H**ierarchies
BN	**B**ayesian **N**etwork
CADS	**C**ommunity **A**nomaly **D**etection **S**ystem
CADS-AD	**CADS** - **A**nomaly **D**etection
CADS-PE	**CADS** - **P**attern **E**xtraction
CBLOF	**C**luster-**B**ased **L**ocal **O**utlier **F**actor
CD	**C**lass **D**ecomposition
CERT	**C**ommunity **E**mergency **R**esponse **T**eam
CF	**C**luster **F**eature
CITD	**C**orporate **I**nsider **T**hreat **D**etection
CMU	**C**arnegie **M**ellon **U**niversity
COD	**C**ontinuous **O**utlier **D**etection
COG	**C**lassification using l**O**cal clusterin**G**
DBN	**D**ynamic **B**ayesian **N**etwork
dIDS	**d**atabase **I**ntrusion **D**etection **S**ystem
DNN	**D**eep **N**eural **N**etwork
EIT	**E**nsemble based **I**nsider **T**hreat
E-RAIDS	**E**nsemble of **R**andom subspace **A**nomaly detectors in **D**ata **S**treams
FN	**F**alse **N**egative
FP	**F**alse **P**ositive
GBAD	**G**raph **B**ased **A**nomaly **D**etection

GBM	**G**radient **B**oosting **M**achine
GLAD-PC	**G**raph **L**earning for **A**nomaly **D**etection using **P**sychological **C**ontext
HMM	**H**idden **M**arkov **M**odel
IADS	**I**nsider **A**ttack **D**etection **S**ystem
iForest	**i**solation **Forest**
IWNB	**I**nstance **W**eighted **N**aive **B**ayes
k-**NN**	k-**N**earest **N**eighbour
LOF	**L**ocal **O**utlier **F**actor
M-B-A	**M**achine learning - **B**ehaviour-based - **A**nomaly detection
M-B-C	**M**achine learning - **B**ehaviour-based - **C**lassification
M-B-R	**M**achine learning - **B**ehaviour-based - st**R**eaming
MCOD	**M**icro-cluster-based **C**ontinuous **O**utlier **D**etection
Meta-CADS	**Meta** - **CADS**
M-Q	**M**achine learning - se**Q**uence-based
MTurk	**M**echanical **T**urk
NSA	**N**ational **S**ecurity **A**gency
ocSVM	**o**ne **c**lass **S**upport **V**ector **M**achine
PHMM	**P**rofile **H**idden **M**arkov **M**odel
PRODIGAL	**PRO**active **D**etection of **I**nsider threats with **G**raph **A**nalysis and **L**earning
RADISH	**R**eal-time **A**nomaly **D**etection system **I**n **S**treaming **H**eterogeneity
RADISH-A	**RADISH** - **A**lerting
RADISH-L	**RADISH** - **L**earning
RandSubOut	**Rand**om **Sub**space **Out**lier detector
RF	**R**andom **F**orest
RNN	**R**ecurrent **N**eural **N**etwork
S	**S**ignature-based
SMO	**S**equential **M**inimal **O**ptimisation
SMOTE	**S**ynthetic **M**inority **O**versampling **T**echnique
STORM	**ST**ream **O**utlie**R** **M**iner
SUTD	**S**ingapore **U**niversity of **T**echnology and **D**esign
SVD	**S**ingular **V**alue **D**ecomposition

SVM	**S**upport **V**ector **M**achine
TN	**T**rue **N**egative
TP	**T**rue **P**ositive
-TRE	**T**rapper **R**emoval
TWOS	**T**he **W**olf **O**f **S**utd
XABA	Zero-Knowledge **A**nomaly-based system **B**ehavioural **A**nalysis
XGBoost	e**X**treme **G**radient **Boost**ing

To those who sacrificed their priorities for me to be here today;

To my parents

Mr & Mrs
Hammad HAIDAR & Hayat AOUAD

Chapter 1

Introduction

1.1 Background

The insider threat problem has historically been one of the most important problems in security; as the ancient Roman satirist Juvenal wrote, "Who will guard the guards themselves?" The situation is no different in the Digital Age [1].

Insider threat detection is a growing challenge for companies and governments trying to protect their assets from malicious insider threats. An insider is a current or former employee, contractor, or business partner of an organisation who has authorised access to the network, system, or data [2] (e.g. strategic or business plans, engineering or scientific information, source code, etc.). We can distinguish between two types of insider threats: (1) malicious, and (2) unintentional. A malicious insider threat refers to a malicious insider who exploit their privileges with the intention to compromise the confidentiality, integrity, or availability of the system or data [3]. An unintentional insider threat refers to an insider who, through action or inaction, causes harm to the confidentiality, integrity, or availability of the system or data [4]. Our research work is primarily concerned with malicious insider threats, where the motivating factors for insiders would be revenge, financial gain, or to the advantage of a competitor company with the promise of a higher-paying job [3]. The Community Emergency Response Team in Carnegie Mellon University (CMU-CERT) [2] identifies three types of malicious insider threats as follows:

- Fraud: unauthorised modification, addition, or deletion of data in an organisation, or identity theft (e.g. credit card fraud);

- IT sabotage: use of Information Technology (IT) to disrupt or halt a business operation in an organisation; and
- Theft of intellectual property (IP): stealing proprietary information from the organisation. This includes industrial espionage involving insiders.

The growing number of insider incidents is generating increased concern from academia, industry, and governments in recent years. The 2018 Insider Threat Report [5], released by the Crowd Research Partners, found that 90% of the organisations feel vulnerable to insider attacks, where 53% of the survey respondents confirmed typically less than five insider attacks against their organization in the past year. Furthermore, the 2018 Insider Threat Report states that 64% of the organisations are focusing on the detection of the insider threats followed by the prevention and post breach forensics.

According to the Software Engineering Institute (SEI) Cyber Minute [6], a recent survey stated that 27% of cyber crime incidents were suspected to be caused by insiders, and 30% of the respondents thought that the damage caused by insider attacks was more severe than the damage from outsider attacks.

The 2015 U.S. State of Cybercrime Survey [7], conducted by CSO Magazine, the U.S. Secret Service, the CERT Division of the Software Engineering Institute, and Price Waterhouse Cooper, stated that 23% (compared to 28% in 2014 [8]) of cyber crime incidents were attributed to insiders. Furthermore, 45% (compared to 46% in 2014 [8]) of the respondents asserted that insider attacks are more costly and damaging to their organisations compared to outsider attacks.

According to the 2011 Cybersecurity Watch Survey [9], 21% of attacks were attributed to insiders in 2011, however, 33% of the respondents viewed the insider attacks to be more costly, compared to 51% of the respondents in 2010.

> *The public may not be aware of the number of insider events or the level of damage because 70% of insider incidents are handled internally without legal action, which is consistent with the 2010 study [9].*

The risk of the malicious activities carried out by insiders is worth greater attention than to outsiders, due to the extent of disturbance caused to an organisation.

TABLE 1.1: A number of malicious insider incidents known worldwide.

Year	Country	Organisation	Insider	Job title	Threat type
2017	USA	NSA	Harold Martin	former contractor	Theft of IP
2015	UK	TalkTalk	Anonymous - Indian	third party contractor	Theft of IP
2013	USA	NSA	Edward Snowden [10]	former contractor	Theft of IP
2012	USA	EnerVest	Ricky Joe Mitchell	former network engineer	IT sabotage
2009	USA	United States Army	Bradley Manning	army soldier	Theft of IP
2008	France	Societe Generale	Jerome Kerviel	trader	Fraud
2007	USA	FIS	William Sullivan	database administrator	Fraud
2004	UK	Royal Bank of Scotland	Donald Mackenzie	royal manager	Fraud
2000	USA	OMEGA	Tim Lloyd	former network administrator	IT sabotage

Most of the security mechanisms implemented in an organisation usually tackle the outsider attacks (e.g. anti-viruses, firewalls, intrusion prevention and detection systems). However, the privileges given to insiders make the detection of the malicious insider threats more challenging. The users within an organisation are aware of the system and have authorised access to sensitive information, which, if disclosed, will result in costly consequences.

In Table 1.1, we go through several examples of malicious insider incidents known worldwide, some of which are reported in the academic literature and others over the media.

Harold Martin was recently convicted for stealing around 500 million pages of national defence information from the National Security Agency (NSA) over the course of 20 years [11]. In 2015, an Indian contractor at the TalkTalk company breached customer data on the mobile sales site [12]. Earlier, Edward Snowden, the famous whistle blower, disclosed 1.7 million classified documents from NSA to newspapers and mass media [13].

Snowden proceeded to hand over countless classified documents, resulting in the biggest leak in the history of the United States [10].

In 2012, Ricky Joe Mitchell reset the servers at EnerVest to their original factory settings, which disrupted the business operations for a month [14]. Earlier, Bradley Manning released to WikiLeaks nearly 750,000 sensitive military and diplomatic documents from the US Army [15]. In 2008, Jerome Kerviel was convicted in the Societe Generale trading loss for forgery and unauthorized use of bank's computers, resulting in loss valued at €4.9 billion [16]. Earlier, it was discovered that William

Sullivan had stolen 2.3 million consumer records from the Fidelity National Information Services (FIS) [17].

The Sullivan case highlighted the threat posed to corporate data and systems by rogue insiders [18].

Donald Mackenzie carried out £21 million fraud loans in the Royal Bank of Scotland over the course of five years up to March 2004 [19]. Early in 2000, Tim Lloyd unleashed a time bomb within the OMEGA software systems, resulting with the loss of \$10 million in revenue and \$2 million in repairing the systems and the lay-off of 80 employees [20]. More incidents are reported with anonymous insiders, and others are handled internally without legal action.

1.2 Problem Statement

There exists a significant body of research that has addressed the insider threat problem, and has proposed a significant number of approaches to detect malicious insider threats. Chapter 2 provides an up-to-date comprehensive survey of the approaches.

Yet still the insider threat problem persists [3].

The approaches proposed for detecting insider threats still have a common shortcoming, which is the great number of false alarms (False Positives – FPs) flagged [21], [22], deceiving the administrator(s) about suspicious behaviour of many users. A false alarm is a result of a normal behaviour detected as anomalous by the system. This maps to normal instances that have a high similarity with anomalous instances, thus appear as suspicious events. The problem of false alarms consumes a valuable time from the administrator's schedule, while investigating the suspected users. On the other hand, it impacts the level of trust between the suspected users and their senior executives in an organisation.

A recent real-time anomaly detection system, named RADISH, based on k-Nearest Neighbours (k-NN) is proposed in [21]. The experimental results show that 92% of the alarms flagged for malicious behaviours are actually benign. We attribute such a high number of FPs to the fact that previously proposed machine learning methods attempt to find all suspicious behaviours (instances) associated with any possible insider threat (i.e. the aim of all-behaviour-all-threat as later described in Section 3.2). However, the focus should be on detecting one or more instance(s) of malicious behaviour(s) per threat (i.e. the aim of any-behaviour-all-threat as later discussed in Section 3.3). Designing machine learning methods with such a relaxing condition, we argue, has the potential to reduce the frequent false alarming problem.

1.3 Aims and Objectives

To tackle the aforementioned shortcoming in the problem statement, we address the insider threat problem in this thesis with the following aims:

- Aim 1: To detect all malicious insider threats, not necessarily all behaviours (instances) that belong to a malicious insider threat (i.e. the aim of any-behaviour-all-threat).
- Aim 2: To reduce the number of false alarms flagged (false positives).

To achieve the aforementioned aims, we set the objectives of our research work that are substantially based on the maturity of the data. We define the maturity of the data in an organisation into three categories: (1) high data maturity, which indicates the presence of labelled data of the normal class and the anomalous class; (2) medium data maturity, which indicates the presence of labelled data of the normal class only; and (3) low data maturity, which indicates the absence of labelled data. The normal class refers to normal behaviours carried out by users in an organisation, while the anomalous class refers to anomalous behaviours that do not conform to the normal baseline of users behaviour. The presence of labelled data of the normal class reveals that an organisation possesses a historical database (data set(s)) of the activities (i.e. actions, commands) executed by users. Furthermore, the presence of labelled data of the anomalous class reveals that an organisation has encountered malicious insider incidents attributed to malicious insider(s). The anomalous class data are often labelled by domain expert(s) and analyst(s) who investigated the malicious incidents.

We incorporate the categories of data maturity to select the suitable approach for insider threat detection. The rationale behind this is to ensure that the machine learning approach is leveraged by all the data class labels available in an organisation. In summary, the objectives of the research are as follows. We propose, develop, and evaluate machine learning approaches that address the different cases of data maturity to meet the aims. The approaches are devised to incorporate procedure(s) (technique(s)) that ensure reducing the false alarms.

- Objective 1: Formulate an opportunistic approach, that addresses the challenges in the existing machine learning approaches, for *any-behaviour-all-threat* detection.
- Objective 2: Address the class imbalance data problem in the case of high data maturity, where there exists minor anomalous instances among the major distribution of the normal class, in a supervised two-class classification approach for detecting malicious insider threats.
- Objective 3: Build a one-class anomaly detection framework that learns a normal baseline model of users' behaviour in the case of medium data maturity to detect malicious insider threats.
- Objective 4: Handle the streaming nature of data in the case of low data maturity in an online anomaly detection approach that ensures in/near real-time detection of insider threats.

1.4 Research Contributions

To meet the aims and the objectives detailed in Section 1.3, we propose the following four contributions which are reported in the following chapters and presented here in summary. As a preliminary step for the machine learning approaches, we define the feature space for the insider threat problem, and we categorise the feature set into five groups. The feature set is extracted from the activity logs of users to construct community behaviour profiles. A community behaviour profile represents the behaviours carried out by a group of users (i.e. community) having the same role.

- Chapter 3: We formulate an opportunistic approach to address the insider threat problem with the aim to detect any-behaviour-all-threat. In the opportunistic approach, it is sufficient to detect any anomalous behaviour in all malicious insider threats.

- Chapter 5: We present the first class imbalance approach for insider threat detection, namely CD-AMOTRE, which combines the Class Decomposition (CD) concept and a novel oversampling technique named Artificial Minority Oversampling and Trapper REmoval (AMOTRE). Class decomposition is presented as an alternative to undersampling the normal (majority) class. It clusters the instances of normal class into subclasses to weaken the effect of the normal class without information loss.

  AMOTRE first removes the minority (anomalous) instances (*trapper instances*) that have a high resemblance with normal instances to reduce the number of false alarms, then it synthetically oversamples the minority class by shielding the border of the majority class. Findings reported in this Chapter are in preparation for publication.

- Chapter 6: We build an anomaly detection framework with two components: one-class modelling component and progressive update component. To allow the detection of anomalous instances that have a high resemblance with normal instances, the one-class modelling component applies class decomposition on normal class data to create k clusters, then trains an ensemble of k base anomaly detection methods, having the data in each cluster used to construct one of the k base models.

  The progressive update component updates each of the k models with sequentially acquired FP chunks; segments of a predetermined capacity of FPs. It includes an oversampling method to generate artificial samples for FPs per chunk, then retrains each model and adapts the decision boundary, with the aim to reduce the number of future FPs. Findings reported in this Chapter have been accepted for publication in [23].

- Chapter 7: We propose a streaming anomaly detection approach, namely Ensemble of Random subspace Anomaly detectors In Data Streams (E-RAIDS), for insider threat detection. E-RAIDS learns an ensemble of p established outlier detection techniques (Micro-cluster-based Continuous Outlier Detection – *MCOD*– or Anytime Outlier detection –*AnyOut*–) that employ clustering over continuous data streams. Each model of the p models learns from a random feature subspace to detect *local outliers*, which might not be detected over the whole feature space.

  E-RAIDS introduces an aggregate component that combines the results from the p feature subspaces, in order to confirm whether to generate an alarm at each window iteration. The merit of E-RAIDS is that it defines a survival factor and a vote factor to address the shortcoming of high number of FPs. Experiments on E-RAIDS-MCOD and E-RAIDS-AnyOut are carried out to test the effectiveness of voting feature subspaces, and the capability to detect [more than one]-behaviour-all-threat in/near real-time. Findings reported in this Chapter have been accepted for publication in [24] and published in [25].

1.5 Ethics Statement

In this thesis, we utilise a collection of **synthetic data sets** generated by the Community Emergency Response Team (CERT) in Carnegie Mellon University (CMU) – CMU-CERT data sets [26] as discussed later in Chapter 4.1. The CMU-CERT repository is an *online open-source* repository that is publicly available for the researchers tackling the insider threat problem. The CMU-CERT data sets log activities executed by *simulated anonymous users* (i.e. not real users). Therefore, these data sets do not hold any confidential user information or content. Hence, the utilisation and preprocessing of these data sets in this work **do not have any ethical implications** that should be considered.

Another aspect is the ethical implications of the Intellectual Property (IP) of this research, including the code implemented for (1) the data preprocessing and feature engineering (Chapter 4), and (2) the proposed and developed approaches(Chapter 5, Chapter 6, and Chapter 7). The code is publicly available on GitHub [27] to ensure the reproducibility of the results. This allows other researchers to run the code, and make comparisons, if needed.

1.6 Structure of the Thesis

The rest of the Chapter is organised as follows. In Chapter 2, we give an up-to-date comprehensive survey of the approaches (machine learning and non-machine learning) to detect insider threats. In Chapter 3, we introduce the opportunistic approach with the aim of any-behaviour-all-threat detection. In Chapter 4, we describe the utilised CMU-CERT data sets, including the malicious insider threat scenarios, and we define, categorise, and extract the feature set in the insider threat problem. In Chapter 5, we present and evaluate the supervised two-class classification approach, namely CD-AMOTRE, to detect insider threats, and address the class imbalance data problem (high data maturity). In Chapter 6, we present and evaluate the adaptive one-class ensemble-base anomaly detection framework for insider threat detection (medium data maturity). In Chapter 7, we present and evaluate the anomaly detection approach, namely E-RAIDS, for in/near real-time detection of insider threats over online data streams (low data maturity). Finally, we conclude the thesis with a summary and future work directions in Chapter 8.

Chapter 2

Approaches for Insider Threat Detection

2.1 Introduction

To our knowledge, the literature presents three survey papers [28]–[30] and a Ph.D. thesis [31] that review the insider threat detection problem. In this work, we give an up-to-date comprehensive survey of the approaches that address the problem.

The insider threat detection problem has been addressed through two different approaches: a signature-based approach which identifies a malicious threat by matching observed instance(s) to patterns (signatures) of anomalous behaviour; and a machine learning approach which learns the baseline of user(s) behaviour so that any deviation from this baseline is identified as anomalous.

We further categorise the machine learning approaches in this review into: a sequence-based approach which detects a sequence of instances (behaviours) as anomalous; a behaviour-based approach which detects an instance (behaviour) as anomalous; and others.

The behaviour-based approaches are further categorised in this review into: a classification approach, which learns from normal and anomalous data (i.e. data maturity is high as defined in Chapter 1) and identifies the decision boundary that separates normal from anomalous; an anomaly detection approach, which learns from normal data only (i.e. data maturity is medium) and detects anomalous instances that deviate from normal instances; and a streaming approach, which learns

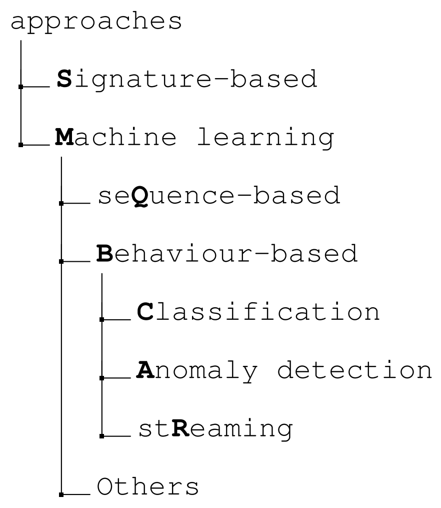


FIGURE 2.1: Categorisation for the state-of-the-art approaches for insider threat detection.

from continuous data streams (i.e. data maturity is low) in order to detect anomalous behaviour. Fig. 2.1 shows the outline for the aforementioned categories of approaches.

2.2 Signature-based Approaches

In the insider threat problem, the principle of a signature-based approach is to detect a malicious insider threat by matching observed instance(s) to patterns (signatures) of anomalous behaviour. This requires a pre-defined database that stores known patterns of insider threats. The known patterns usually refer to previously detected insider threats. However, a signature-based scheme is not able to identify novel (unknown) threats that have not been detected before. Hence, this scheme is sensitive to False Negatives (FNs) (missed unknown insider threats predicted as normal behaviour).

In the following, we give a brief review of the signature-based approaches [32]–[35] proposed in the literature to detect malicious insider threats.

Rose et al. [32] proposed a distributed, automated detection system among endpoint host machines with a centralised repository. Independently, at each endpoint host machine, the critical assets (i.e. data, resources, etc.) and the scenarios of interest are identified. The endpoint system determines the elements that make up the completion of an identified scenario. Based on this, the system tracks the actions of interest only (related to elements of scenario), and determines the likelihood that the scenario elements are completed. The results from endpoints are then pushed into a central repository to aggregate and correlate in the scenario full context, in order to support near real-time alerts. The distributed approach pushes only the results instead of all logs to the central repository, thus reducing the computational and resource consumption. However, as aforementioned, the system is sensitive to FNs.

Agrafiotis et al. [34] address the tripwires, implemented in the Corporate Insider Threat Detection (CITD) system [33], which are responsible for the level '1' alerts. The paper defines the formal language (grammar) for two types of tripwire: policy violations tripwires, which include access to black-listed websites and use of unauthorised USB; and attack pattern tripwires, which capture pre-defined patterns (sequence of actions of interest). If the user's behaviour matches the defined grammar for either of the tripwires, the CITD system generates a level '1' alert. A more detailed description of the whole CITD system, including this signature-based level [34], is given in Section 2.3.2.2.

Ambre and Shekokar [35] present a rule-based event correlation system to detect insider threats. The system is initialised with a configuration of a variety of rules (e.g. rules including a threshold attribute, a suppress condition, a time attribute). The rules define regular expressions to process (match) events, and eliminate the unwanted elements. The system defines an event correlation component that determines the correlation between different events (e.g. previous events). It then utilises Bayesian theorem to calculate the True Positive (TP) rate (conditional probability).

2.3 Machine Learning Approaches

Machine learning, a branch of artificial intelligence, learns the baseline of user(s) behaviour(s), so any deviation from this baseline is identified as anomalous. The anomaly may be detected in two forms: as a single behaviour (instance), or as a sequence of behaviours (instances). Based on this, we categorise the machine learning approaches into: sequence-based, behaviour-based, or others.

2.3.1 Sequence-based Approaches

Some approaches define the feature space in the insider threat problem as an ordered sequence of instances; a sequence of behaviours executed in order of time. A malicious insider threat is identified as a sequence of behaviours in a specific order attributed to a malicious insider.

In the following, we provide a review of the proposed sequence-based approaches.

A recent framework based on a graph approach and isolation Forest (iForest) was presented by Gamachchi et al. [36] to isolate suspicious malicious insiders from the workforce. The graph approach extracts graph and subgraph properties based on user activities, as well as time dependent features to generate input for iForest. iForest then calculates the anomaly scores to separate anomalous behaviour of a user from normal behaviour, without profiling normal behaviour. The framework demonstrates the effectiveness of iForest in identifying malicious insiders, while considering each feature at a time, compared to taking the whole set of features into consideration. The results reveal that more than 79% of users are considered benign (having normal behaviour), while the remaining 21% of users are identified as exhibiting suspicious behaviour (having an anomaly score higher than 0.7).

Furthermore, Gamachchi and Boztas [37] propose a framework based on attributed graph clustering and outlier ranking for insider threat detection. The first step employs a statistical technique to extract the attributes (features) from heterogeneous input data and generate an attributed graph. The next step applies graph clustering to identify the subgraphs (attribute subsapces). Then the vertices (users) of the graph are ranked based on the identified subgraphs. The proposed framework shares some characteristics with the author's framework [36], however, an outlier ranking technique named GOutRank is employed instead of iForest to detect the malicious users. The results report an AUC=0.7648 confirming the effectiveness of the framework.

Moriano et al. [38] suggest a temporal bipartite graph of user-system interactions to detect anomalous time intervals. It is worth noting that the reviewed graph-based approaches detect anomalous structures (e.g. subgraphs), however, the proposed bipartite approach detects anomalous sequence of actions (patterns). The bipartite graph approach compares the number of edges in the user graph to that in the community partition in the aggregated data, so that a sequence can be detected as anomalous or not. The results of the proposed approach outperform a random algorithm, where it reports an F1 measure=85.7%.

Eberle and Holder [39] suggest a graph-based framework to detect malicious insider threats using graph substructures. The framework employs three different versions of the Graph-Based Anomaly Detection (GBAD) algorithm: information theoretic GBAD, which examines instances of the substructures that are similar to normal substructures (i.e. modifications); probabilistic GBAD, which examines all extensions to normal substructure (i.e. insertions); and maximum partial substructure GBAD, which examines parent substructures that are missing various edges and vertices (i.e. deletions). As mentioned in [37], the use of multiple algorithms, each specific for a particular anomaly, complicates the anomaly detection framework.

Huang and Stamp [40] propose Profile Hidden Markov Model (PHMM) compared to the standard Hidden Markov Model (HMM) to detect masqueraders. Masqueraders can be defined as insiders who infiltrate a system's access control to overcome access to a user's account. In contrast to HMM, PHMM relies on positional observations which include a session start and a session end features. This allows it to model the order in which a user issued a sequence of commands based on position-dependent probabilities. The conducted experiments compared PHMM to standard HMM using FN rate with respect to FP rate, showing that HMM slightly outperforms PHMM. However, PHMM shows better performance when no sufficient training data is available. Hence, the proposed PHMM is well suited to be used at the early stages, and HMM afterwards, when sufficient training data is acquired.

Tang et al. [41] suggest a hybrid approach of Dynamic Bayesian Network (DBN) and HMM. DBN models data over session slots and assigns an abnormality score for each user. The higher the score, the higher the malicious threat level. Since the DBN model satisfies the Markov property, the approach transforms the DBN model into an HMM model constructed over an initial state and two session slots. The paper claims that transforming DBN to HMM then applying an HMM inference algorithm shows better performance than applying DBN on its own.

2.3.2 Behaviour-based Approaches

A behaviour-based approach identifies a behaviour (instance) as anomalous or normal. An instance is defined as a feature vector (a set of features) extracted from the activity logs of a user or a group of users. Hence, an anomalous behaviour is a feature vector whose features' values deviate from the normal baseline. In the feature space, this may appear as instances that are located away from the normal instances (majority instances).

As described in Fig. 2.1, a behaviour-based approach is categorised into: classification, anomaly detection, and streaming, based on the maturity of the data. In this Section, we review the existing machine learning approaches in each of these categories, to pave the way for our approaches in Chapters 5, 6, and 7.

2.3.2.1 Classification

In the case of the availability of both normal and anomalous data (i.e., data maturity is high), the system is capable to learn on both classes. Hence, the insider threat detection problem is addressed using a two-class classification approach. In what follows, a review of the two-class classification approaches proposed for insider threat detection is provided.

Mayhew et al. [42] describe a Behaviour-Based Access Control (BBAC) approach that analyses user behaviour at the network layer, the HTTP request layer, and the document layer to detect behaviour changes. BBAC constructs multiple Support Vector Machine (SVM) classifiers, each analyses a certain type of user behaviour in real enterprise data such as, TCP connection behaviour, HTTP request behaviour, and text behaviour, to classify data as normal or malicious insider threat. BBAC combines k-means clustering and C4.5 decision tree in its approach to improve scalability and reduce false positives. The paper reports a variety of results depending on the data type of the malicious behaviour detected, where the TP rate=$76-99.6\%$ and FP rate=$0.18-1\%$.

Punithavathani et al. [43] present an Insider Attack Detection System (IADS) based on the supervised k-Nearest Neighbours (k-NN) to counter insider threats in critical networks. IADS consists of two engines: a network traffic engine that monitors and captures incoming and outgoing packets in traffic, then extracts the important features and stores in a database; and a log analyser engine that mines user logs, extracts patterns of user behaviour, and applies k-NN to identify normal and anomalous patterns based on the closest training patterns in history. In order to verify the detection of malicious threats, IADS employs the computational intelligence to cross check the anomalous patterns (detected by the log analyser engine) with the network features (extracted by the network traffic engine).

Azaria et al. [44] present a Behavioural Analysis of Insider Threat (BAIT) framework that contains bootstrapping algorithms built on top of SVM or Naives Bayes classifiers. The BAIT algorithms include: an SVM classifier with an update procedure using the top k malicious users; an SVM classifier with an update procedure using the top k malicious users and the top $l \cdot k$ benign users; an SVM classifier with an *iterative* update procedure using either a malicious or an honest user one at a time; and a Naive Bayes classifier with an *iterative* update procedure using the recent probabilities. The algorithm with SVM classifier and iterative update procedure reports the best results with F-measure=0.4, and FP=14 (FP rate=7.36%), however, the baseline SVM classifier reports F-measure=0.273, but much lower FP=9 (FP rate=4.73%).

Sen [45] suggests a semi-supervised approach, namely Instance Weighted Naive Bayes (IWNB), with the aim to reduce the FP rate. In this context, IWNB trains the model with labelled train instances, and then updates (retrains) the model with all unlabelled test instances regardless of the label assigned (predicted) for these test instances by Naive Bayes. The authors claim that the shortcoming of relying on the decision of Naive Bayes is that the updated model will not consider the FPs. To address this shortcoming, IWNB considers all unlabelled test instances and assigns weights (probabilities) to them based on their similarity to labelled instances, so that new instances are assigned lower weights than old instances. In this way, the model adapts to the normal change in user's behaviour, which will consequently reduce the FP rate. Although this approach outperforms other existing approaches, including traditional Naive Bayes, as the number of appended FNs increases, it is not convincing to append FNs to the updated user model. The comparison of IWNB to other approach flags false alarms=4.5 for IWNB, compared to false alarms=1.3 in another approach.

Axelrad et al. [46] presented a Bayesian Network (BN) approach to predict insider threats based on psychological features of malicious insiders. The BN model is constructed, such that the nodes represent the categories of the psychological features, and the directed links between the nodes represent the pairwise correlation ratio between the categories (i.e. an estimation of how much a pair of categories are correlated or inversely correlated). The paper presents an experimental approach with a revised BN, where error rate=64.81% and logarithmic loss=1.344 report lower values than original BN (error rate=66.46% and logarithmic loss=1.427). Although the authors claim a considerable improvement, the values of the error rate and the logarithmic loss for the revised BN are still too high.

Barrios [47] develops a multi-level framework, called database Intrusion Detection System (dIDS), to detect malicious transactions. dIDS employs: at level '1' Apriori Hybrid algorithm to check the match between the new transaction and the prior user's transactions logged in the dIDS database; at level '2' Stochastic Gradient Boosting to find the probability of an anomaly; and at level '3' BN to model the data giving a conditional probability of anomaly to a new transaction based on prior user's transactions. A new transaction is assessed at each level in turn. If it is flagged as anomalous, it is appended to the dIDS database and the assessment is terminated, otherwise, it is assessed at the next level. The multi-level approach benefits from the hybrid utilisation of different algorithms, as well as allows to reduce the consuming time when a transaction is flagged anomalous at an earlier level. The probability of detection reported by the framework is =38.38% which is too low.

Kandias et al. [48] compare the performance of text classification approaches for YouTube user comments to detect malicious insiders (users) holding negative attitude towards law enforcement and authorities. The comments are defined into two categories: category (P), which hold negative attitude towards law enforcement and authorities; and category (N), which hold neutral attitude. The text classification aims to classify the comments into category (P) or (N) using: machine learning approach such as SVM, Multinomial Naive Bayes, or Logistic Regression (LR); a dictionary-based approach with terms and phrases related to law enforcement; or a flat-data approach based on Naive Bayes, such that the comments are augmented with user data (public profile information). LR, followed by SVM, achieve a better performance compared to Naive Bayes in terms of precision, recall, F-measure, and accuracy. The dictionary-based approach reveals a significant deviation from LR which indicates the strength of LR in learning the correlations from the training set. However, the flat-data approach indicates the percentage of comments of category (P) a 10% higher than that in LR.

Brdiczka et al. [49] present Graph Learning for Anomaly Detection using Psychological Context (GLAD-PC) to detect threats. GLAD-PC consists of two threads: structural anomaly detection thread which uses graph structure analysis (Random Forest) on network data to separate the normal from the anomalous; and psychological profiling thread that constructs psychological profiles from behavioural data in order to track sudden changes in the psychological features. GLAD-PC then employs a BN to fuse the anomalous data from each of the two threads. After that, a statistical inference method ranks the anomalous instances to decide whether to flag a threat alarm. The results show that the combination of the two threads outperforms applying each thread on its own.

2.3.2.2 Anomaly Detection

In the case of the availability of only normal data (i.e. data maturity is medium), the system is only capable of learning on the majority class (normal class). Hence, the insider threat detection problem is addressed using an anomaly detection approach. In this case, an anomaly detection approach learns the baseline of user(s) normal behaviour, and detects the instances which deviate from the baseline as anomalous instances. There exists a recent research trend towards employing anomaly detection approaches for insider threat detection. iForest [50] is a model-based anomaly detection algorithm that isolates anomalies from the normal instances instead of profiling normal instances.

In the following, we give a brief review of the anomaly detection approaches.

A recent unsupervised ensemble-based anomaly detection system, namely PROactive Detection of Insider threats with Graph Analysis and Learning (PRODIGAL), was presented by Goldeberg et al. [51]; a result of five years work on the insider threat detection problem [52]–[54]. The authors define three diverse types of user-day detectors that are employed in PRODIGAL including, indicator-based detectors which apply outlier detection techniques to feature subsets related to particular activity(ies); anomaly detectors which identify potential anomalies in the whole feature space related to different aspects of the data; and scenario-based detectors which focus on subsets of features and subsets of target users relevant to a particular scenario in order to allow the comparison with peer groups. iForest is configured as one of the anomaly detectors in PRODIGAL to detect unknown malicious insider threats in user activities. Furthermore, PRODIGAL implements an ensemble approach which combines the scores from multiple detectors, such that the consensus about the most anomalous instances with respect to individual detectors is maintained with respect to the ensemble. The ensemble in PRODIGAL achieves an average AUC=0.85, which approaches the average AUC achieved by the best individual anomaly detector (0.88 to 0.89).

Legg et al. [33] present an automated system, called Corporate Insider Threat Detection (CITD), to detect insider threats in an organisation. The CITD defines a data parser module, including an *activity parser* and a *content parser*, which retrieves and parses the data logs for each session. The activity parser appends the activity logs to tree-structure behaviour profiles and extracts an activity feature set for each session. The content parser was discussed in a further paper [55], where it utlises k-means clustering and Principal Component Analysis (PCA) to extract the psychological features based on the contents of the browsed websites, and appends these features to the activity feature set. The CITD then assesses the feature set, for each session, based on three levels of alerts: level '1' policy violations and threat patterns; level '2' threshold-based anomalies, and level '3' deviation-based anomalies. The level '1' alerts correspond to the tripwires addressed in a further paper [34]. The tripwires only fire if the user's behaviour matches the implemented policy violations or threat patterns, thus will not suffer from FPs. This refers to *signature detection*, however, the level '2' and level '3' alerts are in charge of detecting new threats (i.e. *anomaly detection*). The level '3' alerts is in charge of finding the anomaly threshold, which in turn is used for level '2' alerts. The level '2' alerts compare the anomaly score for each session, and triggers an alert based on the predefined threshold. If the alert is an FP, the system's parameters are refined with the aim to reduce upcoming FPs. The authors test the scalability and performance of CITD on a real data set in an experimental paper [56]. The results show that the alerts are generated for 25% of the staff in a multinational organisation. The high FP is related to the significant change in staff's working hours in a multinational organisation. CITD is also evaluated on synthetic CMU-CERT data set with a precision=42% and recall =100%. Besides, the utilisation of a parallel coordinates plot shows 7 of 10 insider threat cases identified clearly.

Zhang et al. [57] apply the Naive Bayes algorithm to file logs with the aim to identify anomalous users based on their probability of interest in file topics compared to that of the community. The approach first categorises the files into predefined topics. It then constructs two types of probabilistic models: a user behaviour model which defines the probability of a user's interest in each predefined topic (based on their file accesses); and a community behaviour profile which defines the probability of a community's interest in a particular topic given another topic (i.e. conditional probability). A user is flagged as anomalous if the user-topic probability is significantly different from a user-community probability, given the user belongs to the community (i.e. users having the same role).

Gates et al. [58] use the structure of the file system hierarchy to measure the level of access similarity of the files, and detect anomalous behaviour based on a predefined threshold. The paper defines access similarity measure techniques including, self score which compares a user's access similarity to a user's historical accesses; and a relative score which compares a user's average score to other users' accesses. These measure techniques are suggested to be used as a feature in the feature vector. The results show that the use of the feature relative score, which compares to others' accesses, allows to attain a lower number of FPs. The results show the ability to detect 80% of the threats with a percentage of FP=2.5%.

Chen et al. [59] introduced Meta-CADS, an extension of the proposed Community Anomaly Detection System (CADS) [60] to detect insider threats in collaborative environments. CADS consists of two components: a Patter Extraction component (CADS-PE) which extracts user-subject relations (patterns) from access logs to infer communities; and an Anomaly Detection component (CADS-AD) which employs the unsupervised k-NN to compare the user-subject relations to the inferred community-subject relations, so that the user's behaviour which deviates significantly from its community is detected as anomalous. Meta-CADS extends CADS where it incorporates the semantics of subjects (subject-category relations) into CADS-PE to infer complex categories using Singular Value Decomposition (SVD); a step before the community inference. The experiments suggest that Meta-CADS performs better than CADS in terms of AUC when the rate of malicious users with respect to benign users is low (0.5%).

2.3.2.3 Streaming Anomaly Detection

In the case of the absence of data (i.e. data maturity is low), this means that the organisation have no previously logged behaviour for users (no historical data logs). Hence, the insider threat detection problem is addressed using a streaming anomaly detection approach.

Few approaches have proposed a streaming anomaly detection approach, with no prior knowledge of user(s) normal or anomalous behaviour [21], [22], [61]–[63]. We now give a description for these approaches.

Bose et al. [21] propose a Real-time Anomaly Detection system in Streaming Heterogeneity, namely RADISH, based on k-NN to detect patterns and anomalies in data streams. RADISH is composed of two processes: a Learning process (RADISH-L) which learns the patterns of normal behaviour in incoming data streams to derive models; and an Alerting process (RADISH-A) which matches the incoming data against the normal patterns in derived models to detect anomalous behaviour. RADISH-A utilises the unsupervised k-NN to compute the anomaly score; the distance to the k^{th} nearest neighbour from the learned data instances. An instance is flagged as anomalous in real-time if its anomaly score is above a predefined threshold. The experimental results show that RADISH detects 50% of malicious sessions, however, 92% of the alarms flagged are actually benign (FPs).

Among the emerging interest in deep learning, Tuor et al. [61] present a preliminary effort to utilise deep learning in an online unsupervised approach to detect anomalous network activity in real-time. The authors presented two models: a Deep

Neural Network (DNN) model which is trained on each acquired instance only once; and a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) architecture which learns an RNN per user such that the weights are shared among all users, however, the hidden states are trained per user. Each model produces ranked user-day anomaly scores to identify the malicious users per day. Note that the anomaly scores produced firstly are quite large, when the model is still learning the normal baseline, and it decreases over time. The results show that DNN and RNN outperform iForest and SVM.

Zargar et al. [62] introduce a Zero-Knowledge Anomaly-Based Behavioural Analysis Method, namely XABA, that learns each user's behaviour from raw logs and network traffic in real-time. XABA is implemented on a big-stream platform without predefined or preprocessed activity logs, to handle high rates of network sessions. It checks for anomalous patterns in real-time and detects diverse types of insider threat scenarios including traitor admin, third party backdoor, credential sniffer, mail spoofing, and foothold hosting. The authors claim that XABA reports a low number of FPs, when evaluated on a real traitor scenario.

One of the ensemble approaches is an ensemble of one class SVM (ocSVM), namely Ensemble based Insider Threat (EIT), proposed by Parveen et al. [22]. The advantage of ocSVM is that it addresses the minority class issue in insider threat problem, by building a model of majority (normal) instances and classifying test instances based on their geometric deviations from the model. However, ocSVM is applicable to static and finite data. The authors suggested an ensemble approach, based on static ocSVM learners, to model a continuous data stream of data chunks (i.e. daily logs). A data chunk is defined as a set of data instances at a day session; user's logged behaviours per day. The EIT acquires data chunks continuously, such that it learns a new model M_s for each received data chunk C_s at a day session s. The EIT maintains an ensemble of a k predefined number of models M, where the ensemble is continuously updated at each day session upon the learning of a new model M_s. The EIT selects the best $k-1$ models from the k models, having the minimum prediction error over the data chunk C_s, and appends the new model M_s. The rationale behind the continuous update of the ensemble models at each day session is to allow the EIT to learn the change in user's behaviour, with the aim to reduce the number of False Positives (FPs) in the upcoming data chunks. It is worth noting that each new *unlabelled* model M_s at a day session s will turn into a *labelled* model at a day session $s+1$; labelled with the predicted labels at a day session s. The proposed ensemble-ocSVM (with updating) is compared to the traditional ocSVM (without updating). The results show that ensemble-ocSVM outperforms the latter, where it reports a higher accuracy=0.76 and almost half the rate of FP=0.24 compared to ocSVM (accuracy=0.58 and FP rate=0.42).

The authors extended their work in a subsequent paper [63], where the ensemble approach is applied to unsupervised Graph-Based Anomaly Detection (GBAD). The results show that ensemble-ocSVM (accuracy=0.71 and FP rate=0.31) outperforms the ensemble-GBAD in terms of FP(accuracy=0.56 and FP rate=0.54).

2.3.3 Others

Other approaches have been proposed in the last decade to address the problem. They are briefly considered here.

Ahmad et al. [64] present a risk assessment methodology, an extension to their previous work [65], to be applied as a first line of defence against insiders. The principle of the presented methodology is to compute the threat score for each user based on the following metrics: F-attributes which defines the physical or electronic privileges of a user's access/modification of an organisation's assets (e.g. resources, information); F-behaviour which defines a user's malicious intents (e.g. user competence, user interactions with honey files, psychological data); and F-vulnerability which defines the vulnerabilities in the user's machine (e.g. firewall or anti-virus is off, open ports on a machine). The results show a manager and a clerk, who are simulated as malicious, to have a high threat score, while a data entry operator has a medium threat score. The reason behind this is that the latter does not have many privileges. The presented methodology does not replace the detection system, however, it complements it to focus on users with a high threat score.

Furthermore, Ahmad et al. [66] introduce a hashing and watermarking application to secure the exchange of medical documents. First, a hash value is computed when a document is accessed, so that the change in the hash value reveals a suspicious access, otherwise a normal access (if the hash value is the same). Second, a watermarking technique is induced to append the user's details to a document whenever it is accessed or created, so that an illegal access or a forged report is detected. The results show the capability of the introduced approach to prevent/detect the simulated malicious scenarios within $1sec$, with a percentage of false alarms between $0-2\%$.

An earlier framework proposed by Ahmad et al. [67] employs a Genetic Algorithm (GA) as an optimisation procedure along with a fuzzy classifier to classify the level of an insider threat with the aim to reduce false alarms. The proposed framework consists of multiple components including: a risk assessment module, which computes a threat score based on different metrics (F-attributes, F-behaviour, and F-vulnerability) as discussed in [65]; a detection system module, which computes a detection score based on data collected from different information sources (e.g. file access logs, email logs, web logs); an offline analysis module, which analyses the past profile of a user; and a fuzzy classifier, which takes as input the outputs of the aforementioned modules to identify the level of threat. The results show the level of threat of simulated scenarios in less than $1sec$, with a percentage of false alarms between $0-8\%$.

2.4 Discussion

We summarise the state-of-the-art reviewed above in Table 2.1, which demonstrates the papers in descending year order. Each recorded paper is associated with: the proposed method(s); the category it belongs to, based on the categorization provided in Fig. 2.1; and the data set(s) utilised to evaluate the proposed method(s).

This Section discusses the literature in terms of (1) the shortcoming of the high number of false alarms (false positives), and (2) the challenges in the existing machine learning approaches. Later, in Section 4.1, we discuss the literature in terms of the data sets utilised to evaluate the approaches.

2.4.1 Shortcoming of High Number of False Alarms

Few of the existing approaches report the performance in terms of false positives. Table 2.2 lists the state-of-the-art approaches that report their results in terms of the

TABLE 2.1: Summary of the state-of-the-art approaches for insider threat detection. The categories of the approaches are:
S:**S**ignature-based,
M-Q: **M**achine learning - se**Q**uence-based,
M-B-C: **M**achine learning - **B**ehaviour-based - **C**lassification,
M-B-A: **M**achine learning - **B**ehaviour-based - **A**nomaly detection,
M-B-R: **M**achine learning - **B**ehaviour-based - st**R**eaming, or others.

Reference	Year	Authors	Method(s)	Category	Data set
[34]*	2017	Agrafiotis et al.	match of patterns	S	NA
[21]	2017	Bose et al.	k-NN	M-B-R	DARPA ADAMS CMU-CERT r2
[36]	2017	Gamachchi et al.	iForest	M-Q	CMU-CERT r4.2
[37]	2017	Gamachchi and Boztas	attributed graph clustering	M-Q	CMU-CERT r4.2
[51]**	2017	Goldberg et al.	anomaly detection (iForest)	M-B-A	DaRPA ADAMS
[33]*	2017	Legg et al.	PCA anomaly detection	M-B-A	CMU-CERT
[38]	2017	Moriano et al.	bipartite graph	M-Q	IBM ClearCase
[32]	2017	Rose et al.	distributed detection	S	NA
[61]	2017	Tuor et al.	DNN RNN	M-B-R	CMU-CERT r6.2
[64]***	2016	Ahmad et al.	risk assessment	Other	simulated
[66]***	2016	Ahmad et al.	hashing watermarking	Other	simulated medical
[56]*	2016	Agrafiotis et al.	experimental paper	Other	multinational organisation
[62]	2016	Zargar et al.	anomaly detection	M-B-R	simulated
[55]*	2015	Alahmadi et al.	PCA kmeans	M-B-A	website
[35]	2015	Ambre and Shekokar	regular expression Bayesian theorem	S	NA
[42]	2015	Mayhew et al.	SVM kmeans C4.5	M-B-C	enterprise
[43]	2015	Punithavathani et al.	k-NN	M-B-C	real-time packets
[65]***	2014	Ahmad et al.	risk assessment	Other	simulated
[67]***	2014	Ahmad et al.	GA fuzzy classifier	Other	simulated
[44]	2014	Azaria et al.	SVM Naïve Bayes	M-B-C	Amazon Mturk
[58]	2014	Gates et al.	anomaly detection	M-B-A	commercial
[45]	2014	Sen	Naïve Bayes	M-B-C	SEA
[68]	2014	Walton et al.	information theory	M-B-R	CMU-CERT r4.2 CMU-CERT r5.2
[52]**	2014	Young et al.	anomaly detection	M-B-A	DaRPA ADAMS
[57]	2014	Zhang et al.	Naïve Bayes	M-B-A	Sogou laboratory corpus
[46]	2013	Axelrad et al.	BN	M-B-C	simulated
[47]	2013	Barrios	BN	M-B-C	transactions
[48]	2013	Kandias et al.	SVM Naïve Bayes LR	M-B-C	YouTube REST API
[69]*	2013	Legg at al.	reasoning	Other	NA
[63]****	2013	Parveen et al.	ensemble-ocSVM ensemble-GBAD	M-B-R	1998 DARPA Intrusion
[54]**	2013	Senator et al.	anomaly detection	M-B-A	DaRPA ADAMS
[53]**	2013	Young et al.	anomaly detection	M-B-A	DaRPA ADAMS
[49]	2012	Brdiczka et al.	graph, BN	M-B-C	online gaming
[59]†	2012	Chen et al.	graph,k-NN SVD	M-B-A	EHR Access Log
[60]†	2011	Chen and Malin	graph,k-NN SVD	M-B-A	EHR Access Log
[39]	2011	Eberle and Holder	GBAD	M-Q	NA
[40]	2011	Huang and Stamp	HMM	M-Q	Schonlau data set
[22]****	2011	Parveen et al.	ensemble-ocSVM	M-B-R	1998 DARPA Intrusion
[41]	2009	Tang et al.	DBN HMM	M-Q	simulated

The asterisk symbols (*) or the dagger symbol (†) cluster authors' teams who are productive in the area of machine learning approaches for insider threat detection.

minimum FP rate (%), or False Alarm rate (%).

TABLE 2.2: The results in terms minimum FP rate (%) or False Alarm rate (%) of the state-of-the-art approaches.

Reference	Year	FP rate (%) or False Alarm rate (%)
[21]	2017	False Alarm rate=**92%**
[42]	2015	FP rate=$0.18-1\%$
[44]	2014	FP rate=4.73%
[58]	2014	FP rate=2.5%
[63]	2013	FP rate=**54%**
[22]	2011	FP rate=**24%**

We notice that a recent anomaly detection approach, called RADISH [21], reports a False Alarm rate=92%, which is extremely high. Though, RADISH is evaluated on insider threat CMU-CERT $r2$ data set [70]. This result sheds light on a persistent shortcoming in the machine learning approaches addressing the insider threat detection problem. An earlier approach proposed by Parveen et al. [63] reports a significant high FP rate=54% for ensemble-GBAD, followed by FP rate =31% for ensemble-ocSVM. This work [63] was an extension of their paper [22], where ensemble-ocSVM reports FP rate=**24%**. Both ensemble-ocsvm and ensemble-GBAD are evaluated on the 1998 DARPA Intrusion Detection data set [71], which is not specifically designed for the insider threat problem.

A significant low FP rate was attained in BBAC [42] and [58], which report FP rate=$0.18-1\%$ and FP rate=2.5% respectively. BBAC is evaluated on real enterprise data, including network flow information, higher-level transport protocols, audit records, and application-level content, in addition to injected synthetic insider threat data labelled by domain experts. However, the approach in [58] is evaluated on real access logs from a commercial source code repository; only source files. Thus, the approach is specifically designed to mitigate the insider threat problem on the level of files only, and therefore it does not generalise as a detection mechanism for the whole network, system, and data in an organisation. The approach in [58] could be utilised as a component in a detection framework.

Similarly, the BAIT approach in [44] reports FP=4.73%, which is quite low, however, BAIT is deployed on Amazon Mechanical Turk (Amazon MTurk) [72], and evaluated on a designed one-person game data. The game data might not include different types of malicious insider threat scenarios, and therefore does not constitute the challenges of variety and complexity in threat scenarios.

To sum up, the existing machine learning approaches still have a common shortcoming, which is the high number of false alarms (FPs).

2.5 Summary

In this Chapter, we gave an up-to-date comprehensive survey of the approaches that addressed the insider threat detection problem, including (1) signature-based approaches, and (2) machine learning approaches, which in turn includes (2.1) sequence-based approaches, (2.2) behaviour-based approaches, and (2.3) others. We discussed the results of the existing machine learning approaches in terms of FPs to shed light on the common shortcoming, which is the high number of false alarms (FPs).

Chapter 3

An Opportunistic Approach for Insider Threat Detection

3.1 Introduction

The review of the state-of-the-art presents machine learning approaches which have shown merit in addressing the insider threat problem in terms of the detection of malicious insider threats, nevertheless, the shortcoming of the high number of false alarms still persists. This asserts the need to formulate a novel approach that mitigates the shortcoming of FPs, whilst detecting the malicious insider threats.

In this Chapter, we discuss the challenges in the existing machine learning approaches, and we then introduce the formulated opportunistic approach, namely *threat hunting*, which will shape the contribution approaches in this thesis.

3.2 Challenges in Existing Machine Learning Approaches

In the following, we distinguish between the two existing categories of machine learning approaches reviewed in Chapter 2: sequence-based; and behaviour-based. A brief description and illustration is given for the each category of approaches to further discuss their challenges.

3.2.1 Challenges in Sequence-based Approach

A sequence-based approach defines an insider threat as a set of behaviours (i.e. actions, commands) executed in a specific order of time by a malicious insider. The

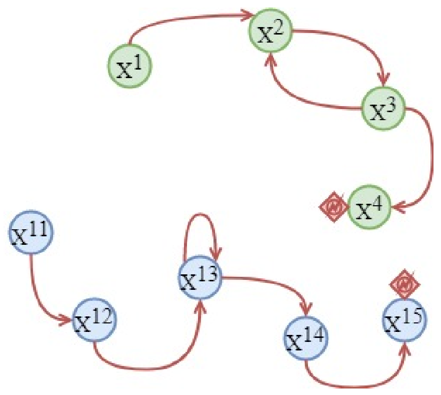


FIGURE 3.1: Sequence-based approach for insider threat detection. Each circle denotes a behaviour X^t executed by a malicious insider at a specific session slot t, such that X^t precedes $X^{t'}$ if $t < t'$. The green circles and the blue circles represent the behaviours which belong to the malicious threats T_1 and T_2 respectively. The red rhombus represents a true alarm of a malicious insider threat. A sequence-based approach will flag a true alarm of a malicious threat T_i, if it detects **all** $X^t \in T_i$ in the given time order.

sequence-based approach models the normal baseline of a user as sequences. It learns the base-line model from the sequences which are often executed, such that a sequence is a set of behaviours executed in a specific order. After that, a set of behaviours, which do not resemble the learned normal sequences, are identified as anomalous, and therefore may refer to a potential insider threat [73].

Figure 3.1 illustrates the concept of a sequence-based approach to detect an insider threat. Each circle denotes a behaviour X^t executed by a malicious insider at a specific session slot (time) t, such that X^t precedes (occurs before) $X^{t'}$ if $t < t'$. For instance, X_1 precedes X_2, indicates that X_1 is executed at a session slot (time) before X_2. Let the green circles represent the behaviours which belong to a malicious threat T_1, and let the blue circles represent the behaviours which belong to a malicious threat T_2. For instance, to detect an insider threat T_1, it is required to detect the green behaviours $X^t, \forall t$ in the given order of time. Let the red rhombus represent a true alarm of a malicious insider threat. A sequence-based approach will flag a true alarm of a malicious threat T_i, if it detects $X^t \in T_i, \forall t$ in the given order.

The drawback of a sequence-based approach when addressing the insider threat problem is that it is essential to detect all behaviours which belong to a threat in the given sequence. The challenge of the insider threat detection problem lies in the *variety* and *complexity* of malicious insider threats in the data sets. A malicious

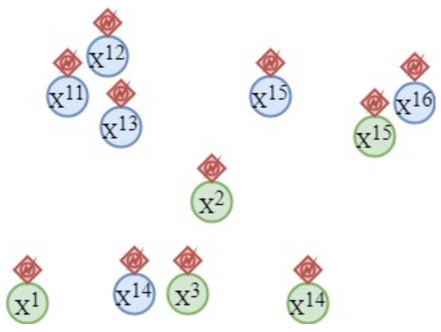


FIGURE 3.2: Behaviour-based approach for insider threat detection. Each circle denotes a behaviour X^t executed by a malicious insider at a specific session slot t. The green circles and the blue circles represent the behaviours which belong to the malicious threats T_1 and T_2 respectively. The red rhombus represents a true alarm of a malicious insider threat. A behaviour-based approach will flag a true alarm of a malicious threat T_i, if it detects **all** $X^t \in T_i$, but regardless of the time order.

insider threat is devised of a complex pattern of anomalous behaviours carried out by a malicious insider, which makes it difficult to detect all behaviours attributed to all malicious insider threats in the given sequence.

3.2.2 Challenges in Behaviour-based Approach

Unlike sequence-based, a behaviour-based approach defines an insider threat as a set of behaviours executed by a malicious insider, regardless of the time order. The behaviour-based approach models the normal baseline of a user from the behaviours which are often executed. The behaviours (instances) which deviate from the normal baseline of a user are considered as anomalous.

Figure 3.2 illustrates the concept of a behaviour-based approach to detect an insider threat. Each circle denotes a behaviour X^t executed by a malicious insider at a specific session slot (time), such that the time order is not important. Let the green circles and the blue circles represent the behaviours which belong to the malicious threats T_1 and T_2 respectively. Unlike the sequence-based, to detect an insider threat T_1, it is required to detect the green behaviours $X^t, \forall t$, regardless of the time order. Hence, it is essential to detect **all** the behaviours which belong to **all** malicious insider threats; the aim to detect all-behaviour-all-threat.

However, as mentioned in the sequence-based approach, the complexity of the malicious insider threat scenarios makes it difficult to detect **all** the anomalous behaviours associated with all threats. The anomalous instances (behaviours), which exist in a dense area of normal instances, have a high similarity to the surrounding normal behaviours. These anomalous instances may be missed by the detection system (False Negatives).

3.3 The Proposed Opportunistic Approach

We notice that the high number of FPs reported in Table 2.2 refers to machine learning approaches (particularly behaviour-based) [21], [22], [42], [44], [58], [63]. We attribute the high number of FPs to the fact that the existing approaches attempt to detect **all** anomalous behaviours in **all** malicious insider threats.

However, the focus should be to hunt the malicious insider who poses a threat to the organisation, hence, to detect the malicious insider threat, but not necessarily all behaviours associated to it.

Given the complexity of the malicious insider threat scenarios and the challenges of the existing approaches, we formulate a novel approach with the aim to detect any-behaviour-all-threat; it is sufficient to detect **any** anomalous behaviour in **all** malicious insider threats. In other words, we can **hunt** a malicious insider threat by at least detecting one anomalous behaviour among the anomalous behaviours associated to this threat. Designing the machine learning approach with such a relaxing condition will contribute in reducing the false alarms. We call this approach *opportunistic*, or sometimes we refer to it as *threat hunting*.

Figure 3.3 illustrates the concept of an opportunistic approach to detect an insider threat. Similar to the behaviour-based approach, each circle denotes a behaviour X^t executed by a malicious insider. Unlike the behaviour-based, to detect an insider threat T_1, it is required to detect **any** green behaviour X^t. Hence, it is essential to detect **any** of the behaviours which belong to **all** malicious insider threats; the aim to detect any-behaviour-all-threat. Note that the opportunistic approach may detect **more than one** behaviour which belong to a malicious insider threat, however, the ultimate aim is to detect a threat for at least one anomalous behaviour.

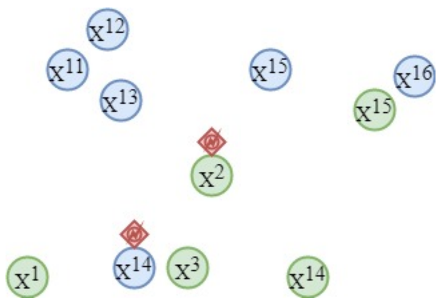


FIGURE 3.3: Opportunistic-based approach for insider threat detection. Each circle denotes a behaviour X^t executed by a malicious insider at a specific session slot t. The green circles and the blue circles represent the behaviours which belong to the malicious threats T_1 and T_2 respectively. The red rhombus represents a true alarm of a malicious insider threat. An opportunistic-based approach will flag a true alarm of a malicious threat T_i, if it detects **any** $X^t \in T_i$.

3.4 Summary

In this Chapter, we discussed the challenges in the existing categories of machine learning approaches which contribute to the shortcoming of false alarms. To mitigate this shortcoming, we formulated an opportunistic approach to shape this research work with the aim of any-behaviour-all-threat detection. In other words, a machine learning approach can detect one or more anomalous behaviours carried out by a malicious insider in order to identify a malicious insider threat. The design of the contribution approaches given this relaxing condition endeavours to reduce the number of false alarms, while detecting all malicious insider threats.

Chapter 4

Feature Space for Insider Threat Detection

4.1 Introduction

A significant impediment to researchers who work on the insider threat problem is the lack of real world data. The real data logs the activities (actions) executed by the insiders (users) in an organisation. These data log files contain: private user profile information (e.g. name, email address, mobile number, home address, etc.); intellectual property (e.g. strategic or business plans, engineering or scientific information, source code, etc.); confidential content (e.g. email content, file content, etc.) [2]. Organisations commonly refuse to give researchers access to real data to protect its users and assets. Under certain regulations, an organisation may agree to grant the researchers limited access, however, after it anonymises the private and confidential attributes in the data.

This issue creates a barrier to the researchers' ability to proceed with their work. Hence, it is preferable, in the insider threat detection problem, to generate/utilise synthetic data for the design and evaluation of the detection system (framework).

Earlier, the research papers utilised a variety of data sets, not specifically designed for the insider threat problem (refer to Table 2.1). For instance, Bose et al. [21] and Goldberg et al. [51]–[54] used the DaRPA ADAMS data set [74]; Parveen et al. [22], [63] used the 1998 DARPA intrusion detection data set [71]; and Huang and Stamp [40] used the Schonlau data set [75]. However, the complexity of the insider threat problem, specifically the complexity of the insider threat scenarios, makes these data sets less relevant. These data sets do not implement malicious insider threat scenarios, and therefore do not well represent the insider threat problem. Hence, the utilisation of these data sets to evaluate the insider threat detection systems is not pertinent.

In the current decade, there is a significant trend towards the utilisation of the CMU-CERT data set(s) for the insider threat detection systems [21], [33], [36], [37], [61], [68] (refer to Table 2.1). The CMU-CERT [26] is a collection of synthetic data sets generated by the Community Emergency Response Team (CERT) in Carnegie Mellon University (CMU). It implements 5 defined insider threat scenarios which are described later in this Chapter.

In the following, we give a description of the utilised CMU-CERT data sets, including the malicious insider threat scenarios addressed in this work. We then define the feature space in the insider threat problem and present the set of features extracted to construct community behaviour profiles (defined later in the following Section).

4.2 Description of the Data Sets

The CMU-CERT data sets are synthetic insider threat data sets generated by the CERT Division at Carnegie Mellon University [26], [70]. The CMU-CERT data repository is the only available data repository that implements malicious insider threat scenarios (5 scenarios) and has recently become the evaluation data repository for researchers addressing the insider threat problem [21], [51], [61].

The CMU-CERT data repository has several data generator releases (including $r1$ to $r6$), such that most releases include multiple versions of data sets (e.g. $r3.1$, $r3.2$). For this thesis, we used $r5.2$ data sets from the insider threat data sets generated by CMU-CERT. This data set logs the behaviour of 2000 employees over 18 months. The rationale behind the selection of the $r5.2$ data sets among the panoply of data generator releases is explained below.

The first step to address the insider threat problem is to determine the model to employ to analyse the behaviour of users in an organisation. We define two types of models: (1) user behaviour model, and (2) community behaviour model.

A user behaviour model defines the behaviour of a single user (employee) in an organisation. It represents only the activities of this user in a structure to analyse the user's behaviour compared to their previous behaviour (only their self).

A community behaviour model broadens and enriches the analysis of a user's behaviour with respect to the community a user belongs to. We define a community as a group of users (employees) in an organisation having the same role. In other words, the users in a community tend to work in a team environment, and the activities required from them are quite similar. Therefore, their behaviours tend to align with the users in the same community. A community behaviour model represents the activities of all the users in the community in a structure to analyse each user's behaviour compared to their previous behaviour as well as to the community's behaviour. In this thesis, we employ the community behaviour models to analyse the behaviour of users and detect malicious insider threats. For instance, consider the community of salesmen, where the activities required from each salesman are to sell and promote commercial products. In the salesmen behaviour model, the activities of the salesmen align, and any deviation of salesman's behaviour from their previous behaviour or the behaviour of the whole community may indicate a malicious insider threat. Hence, we hypothesise that adopting the community behaviour modelling in our work will guide the proposed approaches towards effective detection of malicious insider threats.

Following the definition of a community behaviour model, we justify the selection of the $r5.2$ data sets. Unlike the other released CMU-CERT data sets, the communities in the $r5.2$ data sets consist of a considerable variety of malicious insider threats. In other words, the $r5.2$ data sets implement a considerable number of malicious insiders in each community, such that the scenarios followed by the malicious insider threats in a community are varied. As a result, the richness and variety in the community behaviour models will allow testing of the effectiveness of the proposed approaches and validation of different scenarios. Table 4.1 summarises the released CMU-CERT data sets in terms of the number of implemented malicious insider threats. It is evident that the $r5.2$ release consists of the highest number of malicious insider threats, followed by the $r4.2$ release. Furthermore, the scenarios

TABLE 4.1: Summary of CMU-CERT releases in terms of the number of implemented malicious insider threats.

Release	$r1$	$r2$	$r3.1$	$r3.2$	$r4.1$	$r4.2$	$r5.1$	$r5.2$	$r6.1$	$r6.2$
Number of threats	/	1	2	2	3	70	4	99	5	5

The slash symbol (/) denotes that this release has no implemented malicious insider threats.

implemented in each community of the $r5.2$ release are varied. This is further justified in Table 4.2, where the variety of scenarios implemented in each of the utilised communities is shown.

Among the 2000 employees in the $r5.2$ data sets, we extracted the data logs for the users (employees) belonging to the following three community data sets to be later utilised to validate the proposed approaches:

- Production line worker (com-P): It consists of 300 users, including 17 malicious insiders. It has the scenarios $\{s1, s2, s4\}$ implemented;
- Salesman (com-S): It consists of 298 users, including 22 malicious insiders. It has the scenarios $\{s1, s2, s4\}$ implemented; and
- IT admin (com-I): It consists of 80 users, including 12 malicious insiders. It has the scenarios $\{s2, s3\}$ implemented.

A description of the scenarios $\{s1, s2, s3, s4\}$ implemented in the aforementioned community data sets is in Section 4.2.1.

4.2.1 Scenarios of Malicious Insider Threats

In the following, we give a brief description of the scenarios implemented in the extracted communities:

Scenario $s1$ This scenario considers a user who starts logging in after hours, using a removable drive, and uploading data to the WikiLeaks website. This behaviour occurs for a period of time and the user leaves the organisation thereafter.

Scenario $s2$ This scenario considers a user who starts surfing job websites especially targeting competitor companies. The user's activity of connecting a removable

TABLE 4.2: Summary of the $r5.2$ community data sets in terms of the number of users, the number of malicious insider threats P_T, and the number of malicious insider threats that map to each of the described scenarios $\{s1, s2, s3, s4\}$.

Community	Users	P_T	$s1$	$s2$	$s3$	$s4$
com-P	300	17	6	5	/	6
com-S	298	22	7	9	/	6
com-I	80	12	/	2	10	/

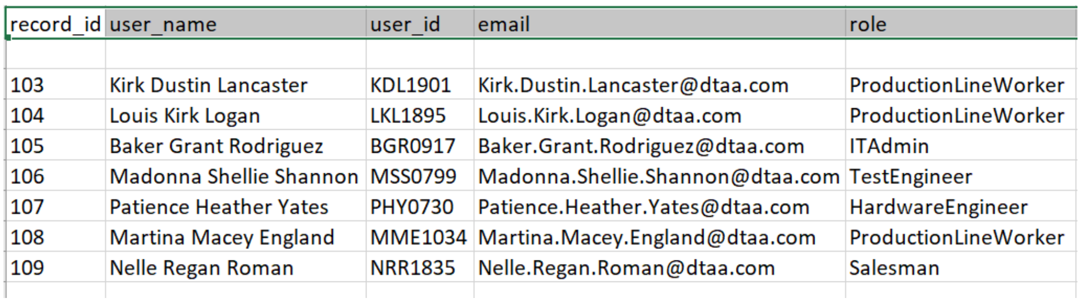
The slash symbol (/) denotes that this scenario is not implemented in the corresponding community data set.

drive to their PC increases incrementally, at a higher frequency than their previous activity, in order to steal data before leaving the company. The activity of *surfing job websites* occurs for a certain period of time, stops for a few session slots, and then reoccurs in a similar manner.

Scenario $s3$ This scenario considers a disgruntled system administrator who downloads a keylogger into a removable drive and connects it to their supervisor's PC to steal their login identity. The next day, the administrator logs into the supervisor's PC using their login identity collected in keylogs and sends an alarming mass email to employees in the organisation. Scenario $s3$ refers to a masquerade insider threat where the malicious insider uses the legitimate user's identity to gain access to their PC.

Scenario $s4$ This scenario considers a user who logs into another user's PC, accesses their files, and emails them to a personal email. It is carried out more and more frequently over a three-month period. New activities (e.g. logon from a new PC, emails to non-employees) occur in a persistent manner to establish a novel behaviour.

More information regarding CMU-CERT data sets and simulated scenarios can be found in [26], [70]. Table 4.2 summarises the aforementioned $r5.2$ community data sets in terms of the number of users (employees), the number of malicious insider threats P_T, and the number of malicious insider threats that map to each of the described scenarios $\{s1, s2, s3, s4\}$.



record_id	user_name	user_id	email	role
103	Kirk Dustin Lancaster	KDL1901	Kirk.Dustin.Lancaster@dtaa.com	ProductionLineWorker
104	Louis Kirk Logan	LKL1895	Louis.Kirk.Logan@dtaa.com	ProductionLineWorker
105	Baker Grant Rodriguez	BGR0917	Baker.Grant.Rodriguez@dtaa.com	ITAdmin
106	Madonna Shellie Shannon	MSS0799	Madonna.Shellie.Shannon@dtaa.com	TestEngineer
107	Patience Heather Yates	PHY0730	Patience.Heather.Yates@dtaa.com	HardwareEngineer
108	Martina Macey England	MME1034	Martina.Macey.England@dtaa.com	ProductionLineWorker
109	Nelle Regan Roman	NRR1835	Nelle.Regan.Roman@dtaa.com	Salesman

FIGURE 4.1: Sample of users' information in CMU-CERT $r5.2$ release.

4.3 Data preprocessing for CMU-CERT Insider Threat Data Sets

The CMU-CERT $r5.2$ release consists of $r5.2$ raw data and users' information. The $r5.2$ raw data comprise system and network logs for the activities carried out by users (employees) in an organisation over 18 months. These activities include: logons, connecting removable drives, browsing urls (websites), copying files, and sending emails. The latter users' information comprises details related to each of the 2000 users (employees) in the organisation. Fig. 4.1 provides a sample of users' information in CMU-CERT $r5.2$ release. It shows the attributes (record_id, user_name, user_id, email, and role) for the records record_id =$103 - 109$. In this work, we are interested in the attribute 'role' which represents a filter attribute to create community data sets. Recall that a community data set consists of the activities carried out by users having the same role (i.e. community).

In this Section, we describe the preprocessing steps we executed to transform the $r5.2$ raw data into community behaviour data sets – 'community matrices'. We used MATLAB $R2016b$ on Windows Server 2016 on Microsoft Azure (RAM $140GB$, OS $64 - bits$, CPU Intel Xeon $E5 - 2673v3$) to implement the data preprocessing procedure. Fig. 4.2 illustrates the preprocessing steps to construct the three community behaviour data sets: Production line worker – com-P, Salesman – com-S, and IT admin – com-I.

For instance, consider the community Production line worker com-P, such that the preprocessing steps for this community are illustrated in blue on the left of Fig. 4.2. The first step selects only the com-P users (employees) from the users' information using the filter attribute 'role'. Second, the $r5.2$ raw data and the selected com-P

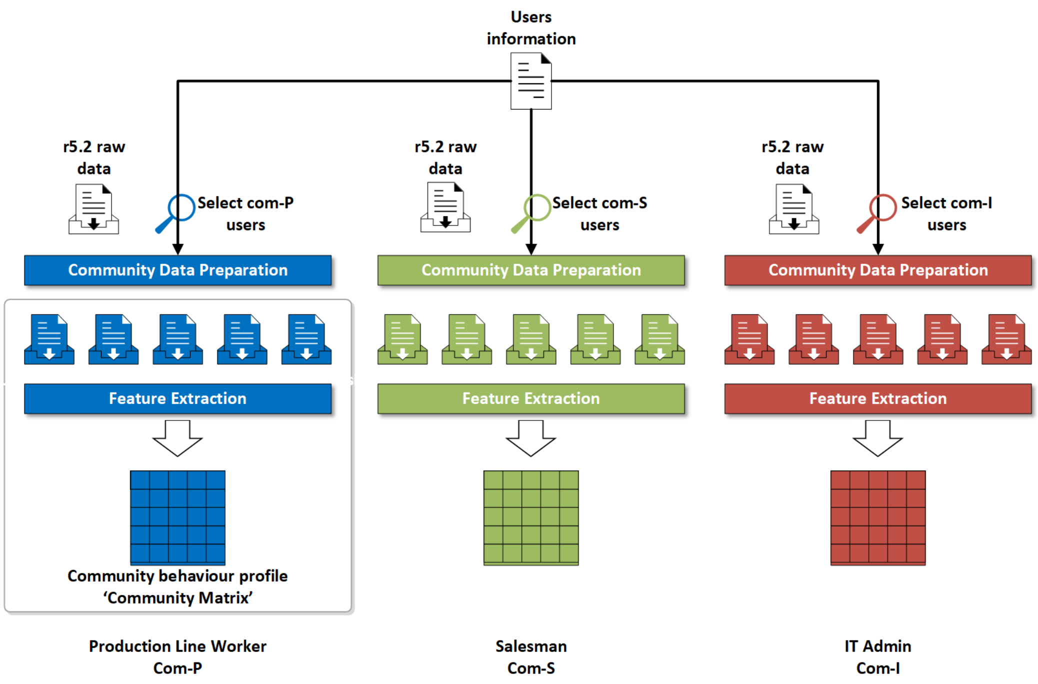


FIGURE 4.2: Data preprocessing of $r5.2$ raw data to construct community behaviour data sets. Each colour illustrates the preprocessing steps of a specific community data set. The block illustrated for the community data set com-P represents the preprocess of 'feature extraction' which is later described in Fig. 4.4.

users are input to the preprocess of 'community data preparation' – preprocess (1). In preprocess (1), the $r5.2$ raw data is filtered using the selected com-P users, and prepared to comprise five documents of activity logs. In each of the documents, we prepare the raw attributes that will be utilised in the preprocess of 'feature extraction' – preprocess (2). The prepared five documents actually refer to five types of activity logs including, logon logs, device logs, http logs, file logs, and email logs. A brief description of the five documents of activity logs is given below.

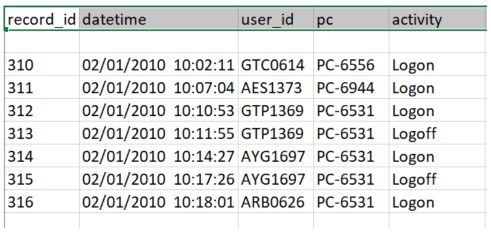
- Logon logs: is a sorted data document that organises the logon activities of the users which belong to com-P. Fig. 4.3a provides a sample of logon logs. It shows the attributes (record_id, datetime, user_id, pc, and activity $=\{$ logon, logoff $\}$) for the records record_id $=310-316$.

- Device logs: is a sorted data document that organises the removable drive activities of com-P users. Fig. 4.3b provides a sample of device logs. It shows the attributes (record_id, datetime, user_id, pc, and activity $=\{$connect, disconnect$\}$) for the records record_id $=795-801$.

- Http logs: is a sorted data document that organises the browsing url activities of com-P users. Fig. 4.3c provides a sample of device logs. It shows the attributes (record_id, datetime, user_id, pc, and url) for the records record_id =$327655 - 327661$. The the attribute 'url' is cut to include only the root domain (hostname).
- File logs: is a sorted data document that organises the file activities of com-P users. Fig. 4.3d provides a sample of file logs. It shows the attributes (record_id, datetime, user_id, pc, and file_extension) for the records record_id =$56599 - 56605$. The attribute 'file_extension' is included instead of the complete name of the file, because the names of the files are simulations of random letters and don't reveal any information. However, the extension of a file reveals the level of importance of a file. For example, executable code files with extension '.exe' have high level of importance. The activity of copying such files to a removable drive would be associated to a malicious insider threat.
- Email logs: is a sorted data document that organises the email activities of com-P users. Fig. 4.3e provides a sample of email logs. It shows the attributes (record_id, datetime, user_id, pc, from, to, cc, bcc, size, attachments) for the records record_id =$992691 - 992697$. For instance, the attribute 'bcc' has a vital role. If the user bcc'd does not belong to the same company (i.e. the domain name of the email address is not '@dtaa.com'), this would be associated to a malicious insider threat. In addition, the attribute 'size' and the attribute 'attachments' would reveal useful information related to the email size and the number of documents attached to the email.

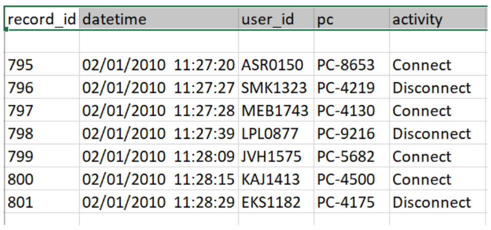
The prepared documents described above are each sorted with respect to the attribute 'datetime' in ascending order. Note that the documents http logs, file logs, and email logs exclude the attribute 'content' from the $r5.2$ raw data. The 'content' is beyond the scope of the thesis, and it requires a text mining add-on. The future directions of this work would consider implementing a text mining add-on for 'content' parsing.

Table 4.3 summarises the five types of activity logs prepared from r5.2 CMU-CERT data for each of the three communities. For the whole $r5.2$ data sets, it gives



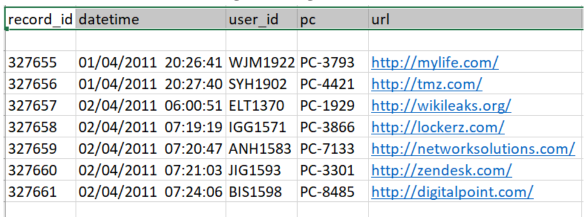
record_id	datetime	user_id	pc	activity
310	02/01/2010 10:02:11	GTC0614	PC-6556	Logon
311	02/01/2010 10:07:04	AES1373	PC-6944	Logon
312	02/01/2010 10:10:53	GTP1369	PC-6531	Logon
313	02/01/2010 10:11:55	GTP1369	PC-6531	Logoff
314	02/01/2010 10:14:27	AYG1697	PC-6531	Logon
315	02/01/2010 10:17:26	AYG1697	PC-6531	Logoff
316	02/01/2010 10:18:01	ARB0626	PC-6531	Logon

(A) Logon logs.



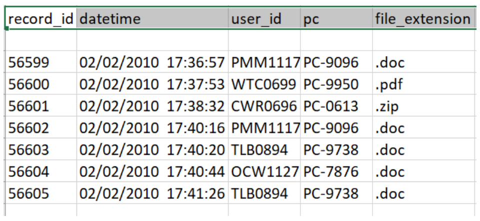
record_id	datetime	user_id	pc	activity
795	02/01/2010 11:27:20	ASR0150	PC-8653	Connect
796	02/01/2010 11:27:27	SMK1323	PC-4219	Disconnect
797	02/01/2010 11:27:28	MEB1743	PC-4130	Connect
798	02/01/2010 11:27:39	LPL0877	PC-9216	Disconnect
799	02/01/2010 11:28:09	JVH1575	PC-5682	Connect
800	02/01/2010 11:28:15	KAJ1413	PC-4500	Connect
801	02/01/2010 11:28:29	EKS1182	PC-4175	Disconnect

(B) Device logs.



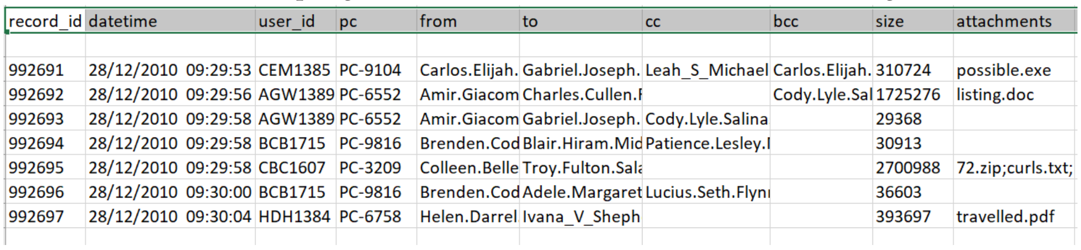
record_id	datetime	user_id	pc	url
327655	01/04/2011 20:26:41	WJM1922	PC-3793	http://mylife.com/
327656	01/04/2011 20:27:40	SYH1902	PC-4421	http://tmz.com/
327657	02/04/2011 06:00:51	ELT1370	PC-1929	http://wikileaks.org/
327658	02/04/2011 07:19:19	IGG1571	PC-3866	http://lockerz.com/
327659	02/04/2011 07:20:47	ANH1583	PC-7133	http://networksolutions.com/
327660	02/04/2011 07:21:03	JIG1593	PC-3301	http://zendesk.com/
327661	02/04/2011 07:24:06	BIS1598	PC-8485	http://digitalpoint.com/

(C) Http logs.



record_id	datetime	user_id	pc	file_extension
56599	02/02/2010 17:36:57	PMM1117	PC-9096	.doc
56600	02/02/2010 17:37:53	WTC0699	PC-9950	.pdf
56601	02/02/2010 17:38:32	CWR0696	PC-0613	.zip
56602	02/02/2010 17:40:16	PMM1117	PC-9096	.doc
56603	02/02/2010 17:40:20	TLB0894	PC-9738	.doc
56604	02/02/2010 17:40:44	OCW1127	PC-7876	.doc
56605	02/02/2010 17:41:26	TLB0894	PC-9738	.doc

(D) File logs.



record_id	datetime	user_id	pc	from	to	cc	bcc	size	attachments
992691	28/12/2010 09:29:53	CEM1385	PC-9104	Carlos.Elijah.	Gabriel.Joseph.	Leah_S_Michael	Carlos.Elijah.	310724	possible.exe
992692	28/12/2010 09:29:56	AGW1389	PC-6552	Amir.Giacom	Charles.Cullen.F		Cody.Lyle.Sal	1725276	listing.doc
992693	28/12/2010 09:29:58	AGW1389	PC-6552	Amir.Giacom	Gabriel.Joseph.	Cody.Lyle.Salina		29368	
992694	28/12/2010 09:29:58	BCB1715	PC-9816	Brenden.Cod	Blair.Hiram.Mid	Patience.Lesley.I		30913	
992695	28/12/2010 09:29:58	CBC1607	PC-3209	Colleen.Belle	Troy.Fulton.Sala			2700988	72.zip;curls.txt;
992696	28/12/2010 09:30:00	BCB1715	PC-9816	Brenden.Cod	Adele.Margaret	Lucius.Seth.Flynn		36603	
992697	28/12/2010 09:30:04	HDH1384	PC-6758	Helen.Darrel	Ivana_V_Sheph			393697	travelled.pdf

(E) Email logs.

FIGURE 4.3: Samples of the five types of activity logs prepared from r5.2 CMU-CERT data for the community com-P.

TABLE 4.3: Summary of the five types of activity logs prepared from r5.2 CMU-CERT data for each of the three communities.

Data set	Logon logs	Device logs	File logs	Http logs	Email logs
$r5.2$	1810070	836984	887621	58960449	17361575
com-P	244458	118441	130656	1029684	464921
com-S	242252	147580	152270	16285182	4241595
com-I	222468	34320	43183	2955296	824248

the number of logs (i.e. records) in each of the activity logs. For each community, it gives the number of logs in each of the activity logs. For instance, the community com-P extracts 244458 logs from 1810070 logs to build the document of logon logs for com-P. In other words, 244458 out of 1810070 logs in $r5.2$ data sets correspond to the logon logs of users who belong to community com-P. In this way, the preprocess (1) of 'community data preparation' prepares the five activity logs for the next preprocess (2) of 'feature extraction'.

4.3.1 Feature Extraction for CMU-CERT Community Data Sets

As aforementioned, the five types of activity logs are prepared with the raw attributes that describe the activities carried out by users who belong to a specific community. The raw activity logs are not susceptible for direct analysis by a machine learning method, however, it is necessary to transform these raw attributes into quantitative features. Fig. 4.2 shows that the activity logs are input to the preprocess (2) of 'feature extraction' and output a 'community behaviour profile' – 'community matrix'.

We define the feature space of the insider threat problem according to the literature [21], [33], [76]. The feature space comprises a set of features that assess the behaviour of users, and allow to compare to previous behaviour of these users or their community of users. We define each feature in the feature set based on the evidence it would give about any undergoing anomalous behaviour. For example, we define the feature logon_after_hours =$\{0, 1\}$. If the value of the feature is 1, then it gives an evidence of an unusual logon activity of a user in the community after the working hours. Therefore, this feature contributes to the overall decision of the system whether an alarm of a malicious insider threat should be flagged or not.

We categorise the features defined in this work into five groups. We give below a brief description of each category:

- Frequency-based '$integer$': assess the frequency of an activity carried out by the users in a specified community during a defined period of time;
- Time-based '$integer$': assess an activity carried out within the non-working hours;
- Boolean '$flag$=$\{0, 1\}$': assess the presence/absence of an activity-related information;
- Attribute-based '$integer$': are more specialised features which assess an activity with respect to a particular value of an attribute; and
- Others '$integer$': assess the count of other activity-related information.

TABLE 4.4: The defined features and their categories.

Feature	Frequency-based	Time-based	Boolean	Attribute-based	Others
freq_logon	x				
logon_after_hours		x			
logon_new_pc			x		
freq_connect	x				
connect_after_hours		x			
freq_browse_urls	x				
freq_browse_job_urls	x			x	
freq_browse_wikileaks_url	x			x	
freq_copy_files	x				
file_access_ext_exe				x	
freq_send_emails	x				
nbr_to_recip					x
nbr_cc_recip					x
nbr_bcc_recip					x
nbr_all_recip					x
non_emp_recip			x		
avg_size_emails					x
nbr_attach					x

The cross symbol (x) denotes that this feature belongs to the corresponding category.

Table 4.4 lists all the features defined in this work and the categories these features belong to. It is worth noting that some features belong to more than one category. For example, 'freq_browse_job_url' is (1) an attribute-based feature, which assesses the activity of browsing particular urls (i.e. urls for job websites), and (2) a frequency-based feature, where it assesses the frequency of executing this activity.

Fig. 4.4 shows the preprocess (2) of feature extraction for a specific community. Let $\{f_1, f_2, \dots, f_m\}$ represent the set of features to construct the community behaviour profile 'community matrix', such that $f_i; 1 \leq i \leq m$ represents a feature from the defined features above. The raw attributes in the prepared activity logs are utilised to extract the values of the features. We define a session_slot as a period of time from start_time to end_time. A community behaviour profile represents a set of feature vectors (i.e. a matrix) over sorted session slots. Each feature vector (i.e. instance) is a set of the values of the features $\{f_1, f_2, \dots, f_m\}$ over a certain session slot. Consider session_slot t, and assume that f_1 is the feature 'freq_connect', then the value of feature f_1 is the frequency of the connect activity for the users who belong to the specified community during the start_time to end_time of session_slot t.

The constructed community behaviour profile is then input to a machine learning approach to generate a community behaviour model. This model defines the

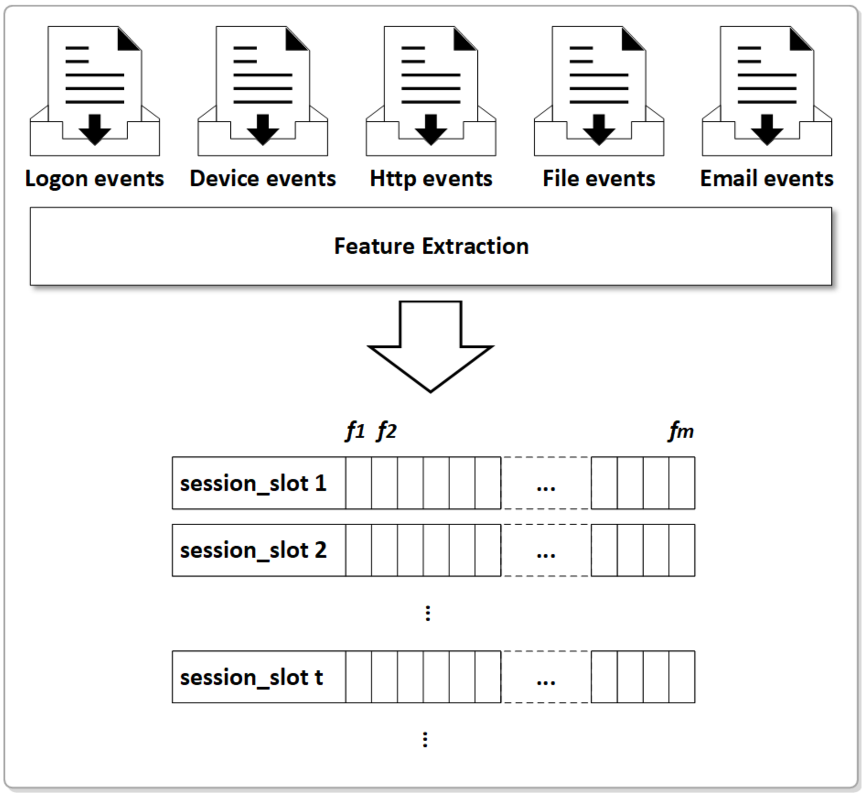
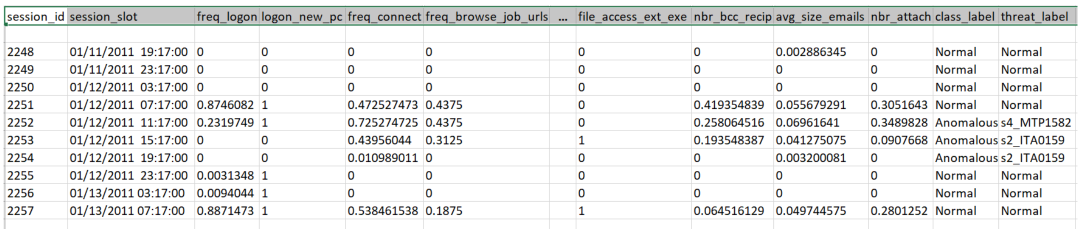


FIGURE 4.4: Feature extraction for a specific community. $\{f_1, f_2, ..., f_m\}$ represent the set of features to construct the community behaviour profile 'community matrix', such that $f_i; 1 \leq i \leq m$ represents a feature from the defined features. A session_slot is a period of time from start_time to end_time. A community behaviour profile represents a set of feature vectors over sorted session slots.

baseline behaviour for the users in the specified community. Any deviation from the baseline is analysed by the detection system to identify whether it is a normal drift of behaviour or it is associated with a malicious insider threat.

In this thesis, based on deep analysis of the data distribution, we define the session_slot per four hours to find local anomalous behaviour within a day which would not be detected per day. The rationale behind choosing the session slot *per four hours* is that this period of time is long enough to extract an instance (i.e. vector of feature values) that provides an adequate evidence of anomalous behaviour. Thus, it allows the system to capture the anomalous behaviours in the feature space. If the session slot is chosen per minutes, for example, the extracted instances would lack adequate evidence of the occurrence of anomalous behaviour. On the other hand, if the session slot is chosen per days/weeks, for example, the period of time



session_id	session_slot	freq_logon	logon_new_pc	freq_connect	freq_browse_job_urls	...	file_access_ext_exe	nbr_bcc_recip	avg_size_emails	nbr_attach	class_label	threat_label
2248	01/11/2011 19:17:00	0	0	0	0		0	0	0.002886345	0	Normal	Normal
2249	01/11/2011 23:17:00	0	0	0	0		0	0	0	0	Normal	Normal
2250	01/12/2011 03:17:00	0	0	0	0		0	0	0	0	Normal	Normal
2251	01/12/2011 07:17:00	0.8746082	1	0.472527473	0.4375		0	0.419354839	0.055679291	0.3051643	Normal	Normal
2252	01/12/2011 11:17:00	0.2319749	1	0.725274725	0.4375		0	0.258064516	0.06961641	0.3489828	Anomalous	s4_MTP1582
2253	01/12/2011 15:17:00	0	0	0.43956044	0.3125		1	0.193548387	0.041275075	0.0907668	Anomalous	s2_ITA0159
2254	01/12/2011 19:17:00	0	0	0.010989011	0		0	0	0.003200081	0	Anomalous	s2_ITA0159
2255	01/12/2011 23:17:00	0.0031348	1	0	0		0	0	0	0	Normal	Normal
2256	01/13/2011 03:17:00	0.0094044	1	0	0		0	0	0	0	Normal	Normal
2257	01/13/2011 07:17:00	0.8871473	1	0.538461538	0.1875		1	0.064516129	0.049744575	0.2801252	Normal	Normal

FIGURE 4.5: Sample of a community behaviour profile 'Community Matrix'.

will be too long to capture the anomalous behaviour blurred among the normal behaviour in the extracted vector of feature values. It is worth noting that early experiments in this work were applied on session slot per minutes, however, the low detection performance guided our work towards session slot per hours. The reason behind this is that the session slot per minutes is too small to carry adequate evidence of the occurrence of anomalous behaviours (as previously mentioned).

After constructing the community behaviour profile, we normalise each vector of feature values (over a session slot) to the range $[0, 1]$, and associate it with a class_label {Normal, Anomalous} and a threat_label {Normal, scenRef_insiderID}. scenRef_insiderID (e.g. s1_ALT1465) has two parts: scenRef (e.g. s1, s2, s3, or s4), which is the reference number for the scenario followed in the malicious insider threat, and insiderID (e.g. ALT1465, AYG1697), which is the user ID of the insider attributed to the threat.

Fig. 4.5 provides a sample of a community behaviour profile 'community matrix'. It shows the attributes (session_id, session_slot), the features (freq_logon to nbr_attach), and the labels (class_label, threat_label) for the session slots session_id =$2248 - 2257$.

4.3.2 The Data Imbalance of CMU-CERT Community Data Sets

An instance is a feature vector over a certain session slot in a community behaviour profile. Hence an instance represents the behaviour of the users in the community during the session slot. An instance is marked as 'anomalous', if it represents malicious activities carried out by the user(s) during the session slot. Otherwise, the instance is marked 'normal'. As discussed in Section 3.3, a malicious insider threat is a threat executed by a malicious insider and associated to a number of anomalous behaviours (instances). We define below the measures N, P and P_T in order to later summarise the data imbalance of the CMU-CERT community data sets:

- N: *Negatives* number of normal instances (normal behaviours);
- P: *Positives* number of anomalous instances (anomalous behaviours); and
- P_T: *Threats* number of malicious insider threats associated to anomalous instances. In other words, P_T is the number of malicious insiders attributed to the anomalous behaviours.

The above defined measures are introduced here only to facilitate the summary of the data imbalance. A complete description of the utilised evaluation measures is provided in the contribution Chapter 5. The defined measures in Section 5.4.3 are further utilised in the other contribution Chapters, nevertheless, the different nature of the problem in each Chapter has required an adjustment to some of the measures. For instance, the streaming nature of the E-RAIDS approach in Chapter 7 required to define an FPAlarm measure, instead of the default (known) FP measure, to evaluate the false positives over a window. A description of the evaluation measures utilised in Chapter 6 and Chapter 7 can be found in Section 6.4.2 and Section 7.4.2 respectively.

Table 4.5 gives a summary of the data imbalance of the community behaviour profiles constructed from CMU-CERT $r5.2$ data. It shows the number of features utilised, the number of instances (i.e. feature vectors) extracted, the number of instances marked normal 'N', the number of instances marked anomalous 'P', and the imbalance ratio 'IR'. IR is the number of instances in the majority (normal) class to the number of instances in the minority (anomalous) class.

It is evident that we are dealing with a data imbalance problem. For example, IR = 22.4545 for com-I, where the N = 2964 is much higher than P = 132. In Chapter 5, we address the class imbalance data problem for insider threat detection.

It is worth recalling that the 17 malicious insider threats in com-P are associated to 366 anomalous instances (behaviours) in the community behaviour profile; the 22 malicious insider threats in com-S are associated to 515 anomalous behaviours;

TABLE 4.5: Summary of the data imbalance of the community behaviour profiles constructed from CMU-CERT $r5.2$ data in terms of the number of features utilised, the number of instances extracted, the number of instances marked normal 'N', the number of instances marked anomalous 'P', and the imbalance ratio 'IR'.

Community	Features	Instances	N	P	IR
com-P	16	3096	2730	366	7.4590
com-S	16	3094	2579	515	5.0077
com-I	15	3096	2964	132	22.4545

and the 12 malicious insider threats in com-I are associated to 132 anomalous behaviours. Moreover, to evaluate the opportunistic approach discussed in Section 3.3, the threat_label is included in the community behaviour profile to validate whether all malicious insider threats are detected.

4.4 Summary

There is a significant research trend towards utilising the synthetic insider threat data sets generated by CMU-CERT for insider threat detection. The CMU-CERT data repository is the only available repository that simulates malicious insider threat scenarios in the data sets. We selected the $r5.2$ release among the panoply of data generated by CMU-CERT to evaluate the proposed approaches in this work. The rationale behind selecting the $r5.2$ release is it richness with malicious insider threats that implement a variety of scenarios within a community.

We select three community data sets to extract from the $r5.2$ raw data which are: Production line worker – com-P, Salesman – com-S, and IT admin – com-I. The $r5.2$ raw data undergoes several preprocessing steps: (1) the users of a specified community are selected from the users' information using the filter attribute 'role', (2) the $r5.2$ raw data and the selected users are input to the preprocess of 'community data preparation' to output five types of activity logs, and (3) the prepared activity logs are input to the preprocess of 'feature extraction'. We define the feature space for the insider threat problem according to the literature [21], [33], [76]. The features are categorised into five groups: frequency-based, time-based, Boolean, attribute-based, and others. The values of the features are then extracted from the different types of raw activity logs to construct community behaviour profiles (e.g. com-P, com-S, and com-I). These community behaviour profiles (data sets) are utilised to evaluate the proposed approaches in this thesis.

The community data sets have a data imbalance ratio, where the number of anomalous (minority) instances is much smaller than the number of normal (majority) instances.

Chapter 5

Supervised Learning for Imbalanced Insider Threat Detection

5.1 Introduction

The class imbalance data problem refers to a data mining problem, where the minority class(es) are substantially under-represented compared to the majority class(es). Several real-world applications pose the class imbalance data problem, in which the data is predominantly composed of 'normal' instances (referring to the majority class) and a small spread of 'anomalous' instances (referring to the minority class) – *two-class imbalance data problem*. Examples of a real-world application include credit card fraud detection, network intrusion detection, and cancer detection. The presence of a 'normal' class and an 'anomalous' class shapes the data mining problem into a classification problem, but with class imbalance. The performance of a machine learning classifier is inevitably biased towards the majority 'normal' class, however, the minority 'anomalous' instances are the interesting examples to learn and eventually detect.

The insider threat detection is a challenging real-world data mining problem. With the scarcity of instances for 'anomalous' behaviours (attributed to malicious insider threats) and the abundance of instances for 'normal' behaviours, we present the insider threat problem as a classification problem with class imbalance. Recall that we are dealing with a *data class imbalance problem* in the insider threat data sets.

For instance, the community IT admin (referred to as com-I) has an imbalance ratio IR = 22.4545, where the number of normal instances N = 2964 is much higher than the number of anomalous instances P = 132 (refer to Table 4.5).

The 'behaviour-based' classification approaches reviewed in Section 2.3.2.1 have shown merit in addressing the insider threat detection problem, however, they do suffer from high false alarms. Some classification approaches report a low FP rate, such as: FP rate=$0.18 - 1\%$ in BBAC [42]; FP rate=2.5% in [58]; and FP rate=4.73% in BAIT [44] (refer to Table 2.2). However, these approaches are evaluated on data sets not specifically designed for insider threats. Thus, their use does not constitute the challenges of variety and complexity in threat scenarios.

To our knowledge, none of the existing approaches addressed the insider threat problem from the perspective of class imbalance, which was discussed in [44]. In this Chapter, we define the insider threat problem as a class imbalance data problem, where the majority class consists of the normal behaviour of a user or a community (i.e. a group of users having the same role), and the minority class consists of the rare anomalous behaviours that map to different scenarios of threats (each threat is typically a number of malicious behaviours/instances). When addressing this problem, we exploited two observations: (1) it is sufficient to detect any anomalous behaviour in any particular threat; and (2) anomalous behaviours resembling normal behaviours contribute to the high false alarms reported in previous work. In the following, we elaborate on each of the aforementioned observations.

As discussed in Section 3.3, we formulate this work with the aim to detect *any-behaviour-all-threat*; it is sufficient to detect any anomalous behaviour in all malicious insider threats (observation (1)).

The anomalous behaviours, in the feature space, often lie within the space of normal behaviours. This appears as anomalous instances having a high resemblance with the surrounding normal instances, which would trap the classifier to detect these normal instances as anomalous (FPs), thus generating a high number of false alarms (observation (2)).

Based on these observations, we propose a hybrid approach, namely CD-AMOTRE, which combines the Class Decomposition (CD) concept and an oversampling technique named Artificial Minority Oversampling and Trapper REmoval (AMOTRE).

Class decomposition is the process of using clustering over the instances of a class or more in a data set to detect subclasses [77]. The process can be applied to all classes or a subset of the classes present in the data. Previous work in addressing class imbalance has considered weakening the effect of the majority class by undersampling its instances [78]–[80] – a process that leads to loss of information, and the possibility of degradation of classification performance as a consequence. Other previous work has considered the idea of clustering to guide the sampling process (referred to as cluster-based sampling). The latter tackles the within-class imbalance and further clusters the minority class before oversampling. However, this process can be ineffective in the insider threat problem due to the scarcity and sparsity of the anomalous behaviours. In this Chapter, class decomposition is used to weaken the effect of the majority class *without information loss* or *cluster-based sampling*.

Synthetic Minority Oversampling Technique (SMOTE) [81] and its variations are the most successful in this area. The process of generating new instances is solely dependent on existing instances in the minority class, or defines the border instances to oversample based on nearest neighbours using all dimensions collectively. In this Chapter, and for the first time, we constrain the generation of new instances when synthetically oversampling the minority class by shielding the borders of the majority class **along each dimension (i.e. feature) separately** in the data set, preventing the generation of instances that may be positioned in close proximity to the instances of the majority class that in turn increases the false alarms. Based on the two aforementioned observations, we devised a method to remove behaviours from the insider threat data (minority class) that have high resemblance with the normal behaviour, and can be a trapper for the classifier to generate false alarms. The removal of such instances can be mostly safe, as based on observation (1), the detection of all anomalous behaviours by the insider is not required.

Specifically, our contributions in this Chapter are as follows:

- a new approach to address the class imbalance problem through class decomposition of the majority and oversampled minority class variations;
- a novel oversampling with selective instance removal technique AMOTRE addressing shortcomings of state-of-the-art methods;

- a detailed analysis of the selective instance removal (trapper removal) on the AMOTRE technique; and
- a thorough performance evaluation of AMOTRE with and without the class decomposition variations (decomposition of the majority class only or both classes) compared to that of SMOTE, showing the superiority of the proposed approaches.

The rest of the Chapter is organised as follows. In Section 5.2, we give a review of the oversampling techniques with a brief description of SMOTE, and a brief background on the utilised supervised learning methods. In Section 5.3, we present the proposed hybrid approach CD-AMOTRE with a formalisation for class decomposition and AMOTRE technique. In Section 5.4, we evaluate the performance of AMOTRE with and without the class decomposition variations compared to that of SMOTE. Finally, we conclude this Chapter with a summary in Section 5.5.

5.2 Background

In this Section, we give a brief review of related oversampling imbalanced data techniques. After that, we give a brief background of the supervised learning methods that are later utilised as base classification methods in the proposed CD-AMOTRE approach.

5.2.1 Oversampling Imbalanced Data

Sampling aims to modify the distribution of a data set to handle the imbalanced data problem. Sampling techniques are grouped into two categories: undersampling, and oversampling. Undersampling (random or selective) is to remove samples (instances) from the majority class. Oversampling is to generate new samples and assign them to the minority class in a way that makes the minority class more dense, thus preventing the bias of the classifier's decision towards the majority class. In the following, we give a review of state-of-the-art oversampling techniques for imbalanced data, because the aim of this work is to propose an oversampling technique to generate artificial samples of the minority class. He et al. [82] proposed ADAptive

SYNthetic sampling technique (ADASYN) which uses density distribution to determine the number of samples to be generated for each minority instance. A leading and widely adopted sampling technique is a hybrid technique, called Synthetic Minority Oversampling Technique (SMOTE) [81]. A brief description of SMOTE is given to introduce the importance of comparing its performance against the proposed approach later in the Chapter.

SMOTE The SMOTE technique was first proposed as an oversampling technique, which introduces artificial samples into the minority class, making it more dense. Let I represent the original set of instances that belong to the minority class, and $perc.over$ be the percentage of oversampling. For each instance $A^t \in I$, SMOTE finds its k_{SMOTE} (parameter k_{SMOTE}) nearest neighbours from the set of minority class instances I. Then, based on the parameter $perc.over$, a $perc.over/100$ number of artificial samples is generated for each A^t. An artificial sample S^t for A^t is introduced along the segment line joining A^t and any of its k_{SMOTE} nearest neighbours (randomly selected). In this way, for each A^t, the artificial samples will be generated along the segments joining any/all of its k_{SMOTE} nearest neighbours. Consider, for example, if $perc.over = 200$, then SMOTE generates $perc.over/100$=2 artificial samples for each A^t. Thus, it introduces $(perc.over/100) \times card(I)$=2 $\times$ $card(I)$ samples into the minority class.

Undersampling was then integrated into the SMOTE technique to remove random samples from the majority class, so that the minority class becomes a specified percentage of the majority class. Let $perc.under$ represent the percentage of undersampling. For instance, if $perc.under = 300$ and the number of artificial samples added to the minority class is 60, then only $(perc.under/100) \times 60$=180 majority instances are randomly selected to remain in the set of majority class instances. The other majority instances are removed. In this way, the SMOTE technique reverses the initial bias of the classifier towards the majority class in the favour of the minority class.

In the following, we describe and argue the suitablilty of some of the techniques proposed as an extension for SMOTE [83]–[86]. Borderline-SMOTE [83] seeks to oversample only the *borderline* minority instances; those which are in the borderline areas (i.e. on or near the decision boundary), where the majority class and the minority class overlap. This technique generates synthetic samples in the neighbourhood of the borderline(s), where the minority instances are most likely to be misclassified. On the other hand, Safe-Level-SMOTE [84] seeks to oversample only the *safe* minority instances; those that are relatively homogeneous areas within the minority class. This technique generates synthetic samples in the areas of safe minority instances in order to strengthen the effect of the minority class. The argument of these techniques is that it only focuses on specific locations (areas) of minority instances to generate synthetic data and disregard the other locations that are also required to be correctly classified.

We argue that Borderline-SMOTE is not suitable for the insider threat problem due to the following. First, this technique assumes a well-defined border(s) of the minority class along all the dimensions (i.e. features), which is not applicable in the insider threat problem. The complexity of the malicious insider threat scenarios manifests in the the high similarity of the anomalous behaviours to normal behaviours. The minority instances (i.e. anomalous behaviours) are typically similar to the majority instances (i.e. normal behaviours), however, at the level of *some* features. In the proposed oversampling technique, we consider the over-sampling at feature level, unlike Borderline-SMOTE. Second, the concept of the Borderline-SMOTE disregards the oversampling of minority instances within the minority class, and therefore the creation of clusters for the minority class is not applicable, making the process of class decomposition less effective.

Hence, in this work, we consider the traditional SMOTE as a more suitable technique to be compared to the proposed AMOTRE oversampling technique, and accordingly, we use it as a benchmark for our experimental study, detailed later in this Chapter.

5.2.2 Classification Methods

In this chapter, we utilise five base methods to address a classification task. These methods include: (1) Random Forest (RF), (2) eXtreme Gradient Boosting (XGBoost), Support Vector Machines (SVM) (3) with linear kernel; (4) with polynomial kernel; and (5) with radial basis function kernel.

The methods were selected based on the experimental studies in the literature on the performance of machine learning classifiers on benchmark data sets, and furthermore, on class imbalance data sets specifically. Fernández-Delgado et al. [87] present an exhaustive evaluation of **179** classifiers over **121** data sets from the UCI machine learning repository [88], including the whole archive except for the large-scale problems to achieve non-data-dependent conclusion on the classifiers. The best results were achieved by *RF*, followed by *SVM with radial basis function kernel*, and then *boosting ensembles*. López et al. [89] present an extensive experimental study of three classifiers including *SVM with polynomial kernel* over **66** imbalanced data sets from the KEEL data set repository [90], [91] to tackle the class imbalance problem. SVMs were concluded as the most robust classifiers over imbalanced data sets. A comprehensive suite of **1,232,000** experiments is given in [92] over **35** benchmark data sets from a wide variety of application domains including **19** imbalanced data sets from the UCI machine learning repository. *RF* and *SVM with linear kernel* were selected for the experiments, and particularly SVM showed a high AUC of an average 0.898 in these experiments.

In the following, we give a description for the methods with the focus on the classification task.

5.2.2.1 Random Forests

Random Forest (RF) [93] (termed Random Decision Forest [94], [95]) is an ensemble of decision trees constructed for a classification or a regression task.

RF was first proposed as a random subspace method, where it combines multiple trees (classifiers) constructed in randomly selected subsets of features (subspaces) [94]. In 2001, Breiman [93] introduced the concept of *bagging* (also called bootstrap aggregating) to extend RFs. Consider a training set $X = \{X^1, X^2, \dots, X^n\}$ of n training instances. Let B represent a predefined parameter for the number of bootstrap trees in RF. Let k_{RF} (parameter k_{RF}) represent a fixed predefined parameter for the number of features (variables) to select a subset (i.e. size of feature subspace). The default value in many implementations is $k_{RF} = \sqrt{m}$ where m denotes the number of features, but $k_{RF} = \log_2 m$ is also used. Below we give a brief formalisation of RF.

- Step 1: RF applies bagging repeatedly for B times to select B random bootstrap samples for the B bootstrap trees, such that each bootstrap sample $S_b; 1 \leq b \leq B$ is selected, with replacement, from the original training set of size m.
- Step 2: For each tree $T_b; 1 \leq b \leq B$, the root node r_b is constructed from the whole bootstrap sample S_b.
- Step 3: At each node $n_{b,i}$, RF (1) selects k_{RF} random features (i.e. feature subset of size k_{RF}), and (2) searches through all the feature subset to find the best split into two child nodes. Unlike traditional pruning of a tree, RF trees are grown all the way down and not pruned. For memory efficiency, a maximum depth or a minimum gain (i.e. improvement) is set in most implementations to stop growing trees very deeply.
- Step 4: To classify a new instance X_t at session slot t, (1) each tree T_b in RF casts a unit vote for a class label, and (2) RF selects the predicted class label y'_t for X_t having the most of B votes.

5.2.2.2 Extreme Gradient Boosting

EXtreme Gradient Boosting (XGBoost) [96] is a scalable gradient boosting library that implements the principles of Gradient Boosting Machine (GBM) [97] for a classification or a regression task.

Gradient boosting is an ensemble of weak base models (decision trees or linear models), where the models are learnt iteratively (i.e. in a boosting fashion), and then generalised by optimising a defined cost function (i.e. loss function). The idea of generalising the models could potentially cause an overfitting issue. To mitigate this issue, several regularisation techniques have been introduced including, (1) *shrinkage* [98], which scales the learnt weights by a factor η after each iteration in tree boosting to reduce the influence of existing trees and allow future trees to improve the model; (2) *subsampling* [99], which implements the concept of bagging proposed by Breiman [100] to fit the model on a subsample at each iteration to prevent overfitting; and (3) *penalising tree complexity* [96], which smooths the learnt weights using the penalty function Ω to minimise the complexity of the model.

Chen and Guestrin [96] describe XGBoost as an ensemble model that learns in an additive manner. Consider a training set of n training instances $D = \left(X^t, y^t\right)$, where X^t represents a training instance at session slot t, and y^t represents the true class label of an instance X^t. The additive function is defined as follows:

$$\hat{y}^t = \sum_{i=1}^{I} f_i(X^t) \tag{5.1}$$

where $i = 1, \ldots, I$ denotes the i^{th} iteration of the total I number of iterations, and $f_i(X)\ \forall i = 1, \ldots, I$ is defined as a classification model (or regression model).

Chen and Guestrin [96] suggest to add f_i at each iteration i to minimise the objective L^t defined as follows:

$$L^t = \sum_{t=1}^{m} l\left(y^t, \hat{y}^t_{i-1} + f_i(X^t)\right) + \Omega(f_i) \tag{5.2}$$

where $\hat{y}^t_i$ represents the predicted class label, and Ω denotes the penalty function to minimise the complexity of the model.

5.2.2.3 Support Vector Machines

Support Vector Machine (SVM) or Kernel Machine [101] (termed Support Vector Network [102]) is a supervised learning method that constructs maximum margin hyperplane(s) to separate labelled data instances for a classification or a regression task.

Linear SVM Consider a data set with two class labels c_1, and c_2. Each data instance (feature vector) X^t at session slot t has a class label $y^t \in \{c_1, c_2\}$. The concept of SVM is to find a *linear* hyperplane (i.e. linear discriminant) to separate the data instances of class label c_1 from the data instances of class label c_2, while maximising the margin. A margin is defined as the distance from the linear hyperplane to the data instances closest to it on either side [101]. These instances are referred to as *support vectors*. In Figure 5.1, we present a two-class classification task, where the blue circles and the red squares represent the data instances of class label c_1 and class label c_2 respectively. Let the solid line represent the *optimal hyperplane* that

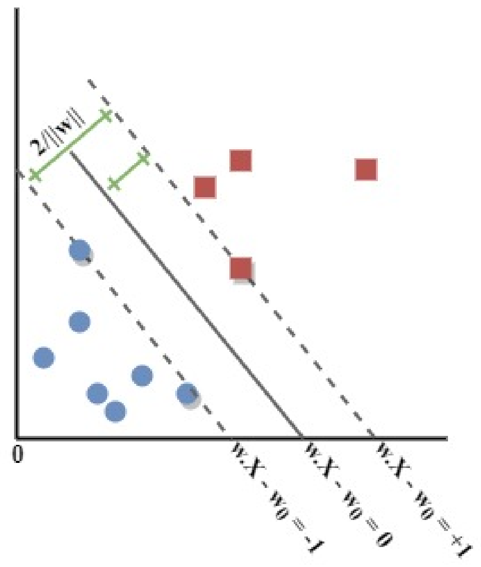


FIGURE 5.1: Two-class SVM classification task. The blue circles and the red squares represent the data instances of class label c_1 and class label c_2 respectively. The solid line represents the *optimal hyperplane* that separates the data instances of the two classes, and the dash lines represent two *margin hyperplanes* that determine the margin. The blue circles and red squares having a shadow represent the support vectors that locate the two *margin hyperplanes* which determine the margin.

separates the data instances of the two classes, and the dash lines represent two *margin hyperplanes* that determine the margin.

The equation for optimal hyperplane is given in Equation 5.3 given the set of instances X satisfying:

$$w \cdot X - w_0 = 0 \tag{5.3}$$

where w is a weight vector that is normal (i.e. normal vector) to the hyperplane; and w_0 is a scalar threshold that determines the location of the hyperplane with respect to the origin 0. Furthermore, the equations for the two margin hyperplanes are given in Equation 5.4 and Equation 5.5:

$$w \cdot X - w_0 = +1 \tag{5.4}$$

$$w \cdot X - w_0 = -1 \tag{5.5}$$

where $+1$ and -1 allow to maximise the margin for the best generalisation. This means that: if $y^t = c_1$ for an instance X^t, then $w \cdot X^t - w_0 \geq +1$ (blue circles); and if $y^t = c_2$ for an instance X^t, then $w \cdot X^t - w_0 \leq -1$ (red squares). As illustrated in Figure 5.1, the blue circles and red squares having a shadow representing the support vectors that locate the two *margin hyperplanes* which determine the margin. As aforementioned, the support vectors are the data instances closest to the optimal hyperplane from either side; the distance from the support vectors to the optimal hyperplane is defined as $\frac{1}{\|w\|}$. In total, the total margin is $\frac{2}{\|w\|}$.

Polynomial SVM and Radial SVM However, the data in a classification task may not be linearly separable; there exists no linear hyperplane (i.e. linear model) that separates two classes. Instead of trying to fit a non-linear model, the feature vectors (data instances) in the original feature space are mapped to a high dimensional feature space using a non-linear basis function (kernel function). Based on the new (mapped) feature vectors, a linear hyperplane is determined. In other words, the linear model in the new feature space replaces a non-linear model in the original space [101]. A kernel function is defined as a measure of similarity over pairs of data instances (X^t,$X^{t'}$) in a data set. In the following, we introduce two types of kernel functions: polynomial, and radial (or Gaussian).

The *polynomial kernel* function is given in Equation 5.6:

$$K(X^t, X^{t'}) = ((X^{t'})^T \cdot X^t + 1)^q \tag{5.6}$$

where X^t and $X^{t'}$ are data instances at session slots t and t' respectively; ^{T} designates a transpose function; and q designates the degree of polynomial.

The *radial kernel* function is given in Equation 5.7

$$K(X^t, X^{t'}) = exp\left[-\frac{\|X^t - X^{t'}\|^2}{2s^2}\right] \tag{5.7}$$

where exp designates an exponential function; s designates a covariance parameter (i.e. spread of values). Note that the larger the s, the smoother the discriminant.

Optimisation Methods for SVM To train the SVM classifier, the classification task is defined as an optimisation problem with the aim to maximise the margin and to minimise $\|w\|$ (see Cortes and Vapnik [102] for more details). Campbell [103] provides a survey of commonly used optimisation methods to solve an SVM optimisation problem including: column generation methods for linear optimisation problems; conjugate gradient and primal-dual interior point methods for quadratic optimisation problems; chunking method; Sequential Minimal Optimisation (SMO) method; and the Lagrangian method.

5.3 CD-AMOTRE: Class Decomposition with Artificial Minority Oversampling and Trapper Removal

The imbalance of data weakens the performance of a supervised classifier in both the learning phase and the prediction phase. In the learning phase, this hinders the classifier from finding an optimal decision boundary to separate between the majority class (normal region) and the minority class (anomalous instances). When a classifier tries to predict the class label of a new instance, it faces two different challenges: classifying a minority instance as majority (i.e. false negatives), or classifying a majority instance as minority (i.e. false positives). False negatives may be attributed to the location of the minority instance in a cluster of majority instances, or to the density of the minority instances within the cluster. On the other hand, false positives may be caused by the similarity between the feature value of a majority instance and the neighbour minority instances.

We address the imbalanced data problem in a hybrid approach, namely CD-AMOTRE, which combines class decomposition and AMOTRE oversampling technique. The ultimate aim of this approach is to detect any-behaviour-all-threats – *threat hunting* (based on observation (1) in Section 5.1), and to reduce the number of false alarms (based on observation (2) in Section 5.1). As stated before, the problem of false alarms is still a limitation in the existing approaches for insider threat detection.

In the following, we present our hybrid approach CD-AMOTRE that comprises two approaches: class decomposition, and the proposed AMOTRE oversampling technique.

5.3.1 Class Decomposition

As described previously, the availability of data of both 'normal' class and 'anomalous class' (i.e. high data maturity in an organisation) shapes the insider threat problem as a supervised classification problem. However, the challenge here lies in the data with high imbalance ratio, where the normal instances dominate the minor number of anomalous instances (i.e. malicious insider threats). The performance of a classification method typically tends to decline when the data distributes in an imbalanced way. One way to handle this problem is to adjust the original region of the class in order to achieve better fit of the class's decision boundary. For example, the use of a polynomial kernel with Support Vector Machine (SVM) uses the original features to project into a polynomial feature space, which allows a non-linear hyperplane to separate classes. However, this would lead to overfitting of the training data, resulting in a poor prediction performance [77]. Overfitting refers to a model (here a classification model) that learns the noise in the training data, and fails to generalise (the problem of high variance), thus degrades the prediction performance [101].

Vilalta et al. [77] proposed the idea of class decomposition to address the problem of high bias and low variance in the classification methods. The idea of class decomposition tackles the class that distributes in a complex way and applies clustering to this class to decompose it into multiple clusters, thus identifying local patterns within the class. The data of the original class label is assigned the corresponding cluster label as a preprocessing step for the classification method. In this way, the classifier learns multiple decision boundaries (per cluster), rather than a single decision boundary with respect to the original class, and thus avoids data overfitting.

In this work, we adopt class decomposition to address the problem of class imbalance data. Although class decomposition was originally proposed to reduce high bias in classifiers [77], it has the property of weakening the effect of the decomposed class when constructing the classification model. Such a property is useful if class decomposition is applied to the majority class to address the class imbalance problem. The idea is to decompose the majority class into clusters (i.e. subclasses) to weaken the effect of the majority class with respect to the minority class.

Some of the previous work addressing the class imbalance has considered weakening the effect of the majority class by undersampling its instances [78]–[80]. As previously mentioned, undersampling removes samples (instances) from the majority class in a way to balance the class distribution of data. In the insider threat problem, the removed instances may hold a substantial knowledge about the normal behaviour of users, and thus will not be represented in the classification model. Hence, the process of undersampling poses a drawback of the loss of information related to the majority (normal) class, which would degrade the performance of the classification method. In this work, the proposed idea of class decomposition is considered to weaken the effect of the majority class, however, retains all the normal behaviours (instances) without information loss.

Other previous work addressing the class imbalance has considered the idea of clustering (referred to as class decomposition) to guide the sampling process – *cluster-based sampling* [104], [105]. The idea is not only to consider the *between-class imbalance*, but also the *within-class imbalance*. The between-class imbalance refers to the imbalance in the number of instances that belong to each class. The within-class imbalance refers to the imbalance in the subclusters within a class. Japkowicz and her co-authors have focused on the within-class imbalance [106]. For instance, the idea in [104] is to cluster each class separately into subclasses, then randomly oversample the instances of the small clusters, so that all clusters will comprise an equal number of instances. In this work, we focus on the between-class imbalance, not the within-class imbalance. The challenge in the insider threat data sets lies in the scarcity and sparsity of the anomalous behaviours attributed to the malicious insider threats. As later discussed in this Chapter, the minority class is difficult to be clustered given its original size (i.e. number of anomalous instances – 'scarcity') and its data distribution (i.e. spread of anomalous instances – 'sparsity'). Instead, we proposed a selective oversampling technique for the minority class as described in Section 5.3.2. A further clustering can be applied to the minority class, however, after oversampling its instances (a discussion in Section 5.3.1.2 is provided). On the other hand, we adopted class decomposition for the majority class only to weaken its effect and determine the local patterns, but no sampling process is applied for it.

A further example is the cluster-based undersampling approach suggested by

Varassin et al. [107]. Its idea is to cluster the majority class into clusters, and then select only the centroids of the clusters to represent the majority class. In this way, the instances of the majority class are removed, and only the centroids of the clusters are kept in the training data. Again, such an approach leads to the loss of information related to normal behaviours.

k-means Clustering for Class Decomposition We select k-means clustering [108] to decompose the class into k clusters (subclasses). k-means clustering is known to be fast, robust, and less computationally demanding. It only requires tuning the parameter k which controls the number of clusters. Favourably, it does not require, and therefore is not influenced by, data-dependent parameters.

k-means clustering [108] is an unsupervised machine learning method that partitions a set of instances into k disjoint clusters, where each instance belongs to the cluster with the nearest centre. The objective of k-means is to minimise the within-cluster sum of squares; a cluster with a small sum of squares is more compact.

k-means first selects k initial centres for clusters (randomly). Second, it assigns each instance to the cluster with the nearest cluster centre – assignment step. The nearest centre is commonly defined in terms of the Euclidean distance. The Euclidean distance between an instance X^t and the centre of a cluster $C^i; 1 \leq i \leq k$ is defined as follows:

$$dist(X^t, C^i) = \sqrt{\sum_{f=1}^{m} \left(X_f^t - C_f^i\right)^2} \tag{5.8}$$

where $X_f^t; 1 \preceq f \preceq m$ represents the value of the f^{th} feature of instance X^t, and $C_f^i; 1 \preceq f \preceq m$ represents the value of the f^{th} feature of cluster centre C^i.

k-means then updates the centre of each cluster based on the instances assigned to it – update step. The steps of 'assignment' and 'update' are repeated until the method converges, i.e. the centres or the clusters no longer change [109]. Thus, the minimum within-cluster sum of squares is achieved.

There is a considerable body of literature that has successfully applied k-means clustering for class decomposition and/or class imbalance data [104], [105], [109]–

[111]. A recent approach [110] presented an Imbalanced Clustering (IClust) algorithm that uses k-means to identify a set of initial clusters (group structures) in high-dimensional media data.

Banitaan et al. [109] employed k-means clustering and hierarchical clustering to study class decomposition prior to Naive Bayes classification. For k-means, k=2 provided the best results for most of the data sets.

Wu et al. [111] proposed Classification using lOcal clusterinG (COG) in multi-label data with high class imbalance. COG applies k-means clustering in each large (i.e. majority) class, but not across classes, to produce subclasses with relatively balanced sizes – *local decomposition*. COG then applies traditional supervised machine learning methods, including SVM and C4.5 decision tree [112].

Japkowicz, who focused on class decomposition to tackle the within-class imbalance (as aforementioned), used k-means to determine the inner distribution of a class followed by a proposed rebalancing method [105]. In a further approach [104], Jo and Japkowicz suggested applying k-means clustering independently to majority class and minority class, followed by oversampling all the clusters – *cluster-based oversampling*. The best results were obtained at k=4 on each class.

5.3.1.1 Decomposing the Majority Class

As described above, we adopt the idea of class decomposition in the insider threat problem to mitigate the bias towards the majority (normal) class in the classification. We apply k-means clustering method to decompose the majority class into k clusters (subclasses). In this way, the two-class classification problem ('normal' class label versus 'anomalous' class label) is transformed into a multi-class classification problem (labels of clusters (sub-classes) of 'normal' class versus 'anomalous' class).

Hence, class decomposition allows the classification method to delineate multiple decision boundaries instead of one decision boundary and to achieve better separation between the majority subclasses and the minority class, thus improving the prediction of new instances.

Let $X^t = \{x_1^t, x_2^t, ..., x_m^t\}$ represent the feature vector at session slot t, where $x_f^t; 1 \preceq f \preceq m$ represents the value of the f^{th} feature. Let $Y = \{y^1, y^2\}$ represent the

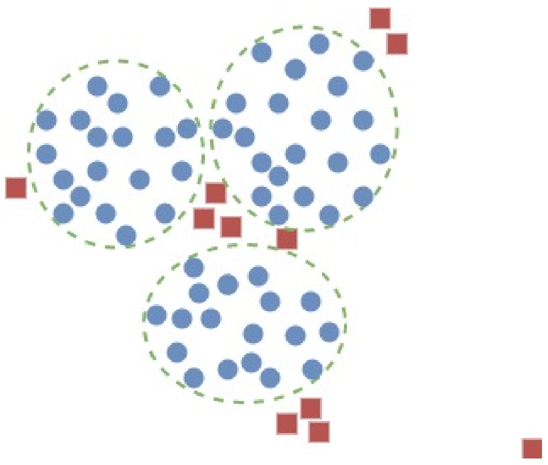


FIGURE 5.2: Decomposition of the majority class into k=3 clusters. The blue circles represent the majority instances (i.e. normal behaviours), and the red squares represent the minority instances (i.e. anomalous behaviours). The green dashed circles represent the k=3 clusters identified among the majority data.

output space, where y^1 is the majority class label and y^2 is the minority class label. Each instance (i.e. feature vector) X^t belongs to a class label $y^j, j = \{1, 2\}$.

Let $M = X^t \ \forall t; X^t \in y^1$ represent the set of instances that belong to the majority class y^1, and let I represent the set of instances that belong to the minority class y^2. If we apply k-means clustering method on the set M, then the class label y^1 will break down into k cluster labels. Hence, each instance X^t in the set M will be assigned a cluster label instead of a class label. Let $\{yc_1^1, yc_2^1, ..., yc_k^1\}$ represent the set of cluster labels belonging to class label y^1, where $yc_c^1; 1 \preceq c \preceq k$ represents the c^{th} cluster label.

Figure 5.2 illustrates the idea of applying class decomposition on the majority class. Let the blue circles represent the majority instances (i.e. normal behaviours), and let the red squares represent the minority instances (i.e. anomalous behaviours). The green dashed circles represent the k=3 clusters (patterns) identified among the majority data. In this case, the problem is defined as a multi-class (*four-class*) classification problem, where the output space is represented as $Y = \{yc_1^1, yc_2^1, yc_3^1, y^2\}$, such that $YC = \{yc_1^1, yc_2^1, yc_3^1\}$ denotes the cluster labels, and y^2 denotes the minority class label.

5.3.1.2 Decomposing the Minority Class

As previously mentioned, we proposed an oversampling technique to tackle the minority class. As later explained in the experiments, the concept of class decomposition may be also applied to the minority class, however, after oversampling its instances. The original size of the minority class is very small, where the number of minority instances (e.g. 132 anomalous behaviours in com-I) is much lower than the number of majority instances (e.g. 2964 normal behaviours in com-I) – *we refer to this as 'scarcity'*. Also, the distribution of minority instances is dispersed, where the minority instances may exist among the majority behaviours or may be dispersed among the whole data – *we refer to this as 'sparsity'*. The scarcity and sparsity of the minority instances in the insider threat data sets makes it difficult to cluster the original minority class I. If we oversample the set I, then the updated set I will append the artificial set I_s. Eventually, the updated set I can then be decomposed into k clusters. We clarify this later in Section 5.3.2.2 (*Generating Artificial Samples at Instance Level*).

Consider $I = X^t \; \forall t; X^t \in y^2$, the set of instances that belong to the minority class y^2. If we apply k-means clustering method on the set I, then the class label y^2 will break down into k cluster labels. Hence, each instance X^t in the set I will be assigned a cluster label instead of a class label. Let $\{yc^2_1, yc^2_2, ..., yc^2_k\}$ represent the set of cluster labels belonging to class label y^2, where $yc^2_c; 1 \preceq c \preceq k$ represents the c^{th} cluster label.

5.3.2 AMOTRE Oversampling Technique

Here, we tackle the class imbalance data problem by sampling the instances of the minority class to modify the original distribution of data among the classes and to achieve an approximate balance between the majority class (or clusters of the majority class) and the minority class. We propose a selective oversampling technique, namely AMOTRE, as an alternative to the state-of-the-art SMOTE sampling technique. In the following, we describe the steps of the AMOTRE oversampling technique.

Algorithm 1 Identifying peculiarity of minority instances

```
1:  foreach $A^t \in I$ do
2:      compute $perclof_M^t$
3:      compute $perclof_I^t$
4:      if $perclof_M^t < \tau$ then
5:          remove trapper instance $A^t$ from $I$
6:          $I_r \leftarrow I \setminus A^t$
7:      end if
8:  end for
9:  foreach $A^t \in I_r$ do
10:     $p^t = \left(\lceil perclof_M^t \rceil \times \lceil perclof_I^t \rceil\right) / 10^4$
11:     $p \leftarrow p \cup p^t$
12: end for
13: return $p$,$I_r$
```

5.3.2.1 Identifying the Peculiarity of Minority Instances

The first step in the AMOTRE oversampling technique is to identify the peculiarity of the minority instances. This guides the process of generating artificial samples for each minority instance A^t in the minority class I. Algorithm 1 gives a brief formalisation for identifying the peculiarity of the minority instances.

LOF for Minority Instances We utilise the Local Outlier Factor (LOF) [113] method to find the LOF for each minority instance $A^t \in I$. LOF is a density-based method that calculates the local outlier factor (score) for each instance in a data set with respect to its k_{LOF} (parameter k_{LOF}) nearest neighbours from the whole data set. For instance, consider that LOF is normalised in the range $[0, 1]$. If the location of an instance is in a high-density region (cluster), then the LOF is closer to 0 (too low). However, if the instance is in a low-density region, then the LOF is closer to 1 (too high). The advantage of LOF compared to global outlier methods is that LOF can identify local outliers in certain regions of the data set which would not be identified as outliers with respect to the whole data set.

We introduce two modified versions of LOF to utilise in the approach proposed in this Chapter: lof_M^t and lof_I^t. In both versions, the idea is to calculate the local outlier factor for each minority instance $A^t \in I$ in the training data set. lof_M^t and lof_I^t are tuned for different values of k_{LOF}. We define lof_M^t and lof_I^t as follows:

Definition 5.3.1: $\boldsymbol{lof^t_M}$ The lof^t_M is the LOF for a minority instance A^t with respect to the k_{LOF} nearest neighbours from the majority class instances (M) only excluding the other minority class. In lof^t_M, we select the value of $k_{LOF} = \sqrt{1 + card(M)}$ (thumb-rule), where the radicand $1 + card(M)$ represents the number of instances utilised to calculate lof^t_M for A^t (1 minority instance A^t + all of majority instances M).

Definition 5.3.2: $\boldsymbol{lof^t_I}$ The lof^t_I is the LOF for a minority instance A^t with respect to the k_{LOF} nearest neighbours from the minority class instances ($I \setminus A^t$) only excluding the majority class. In lof^t_I, we select the value of $k_{LOF} = \sqrt{card(I)}$ (thumb-rule), where the radicand $card(I)$ represents the number of minority instances utilised to calculate lof^t_I for A^t (all minority instances I including A^t).

Note that lof^t_M is still calculated with respect to the whole majority class M (i.e. all the majority instances) regardless of whether class M is decomposed into clusters (subclasses). The reason behind this is that class decomposition only improves classification (from *two-class* to *multi-class* classification) to weaken the effect of the majority class. Similarly, lof^t_I is calculated with respect the the whole minority class I (i.e. all other minority instances) regardless if class I is decomposed into clusters (subclasses).

In the following, we infer the degree of outlierness of a minority instance A^t based on the values of both lof^t_M and lof^t_I in three different cases illustrated in Figure 5.3:

- $high\ lof^t_M$ (close to 1) and $low\ lof^t_I$ (close to 0) $\Rightarrow$ A^t is in a high-density region of minority instances away from the majority instances.
- $high\ lof^t_M$ (close to 1) and $high\ lof^t_I$ (close to 1) $\Rightarrow$ A^t is in an outlier away from the majority and minority instances. We call A^t here an *extreme outlier* instance.
- $low\ lof^t_M$ (close to 0) $\Rightarrow$ A^t is located in a high-density region of majority instances.

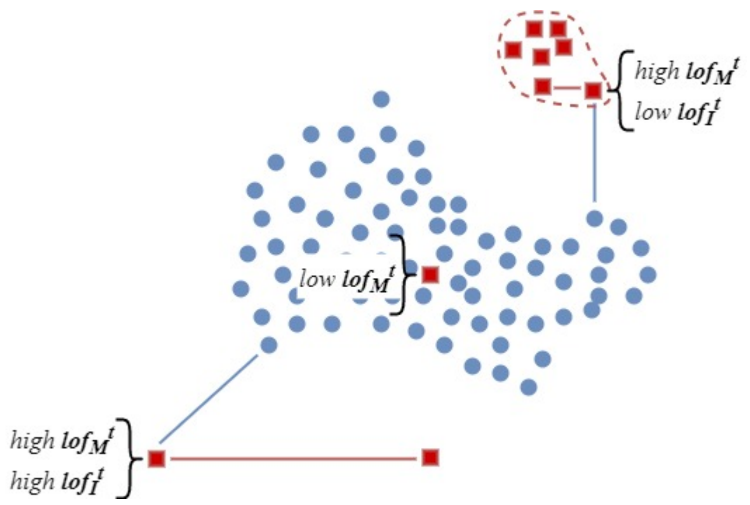


FIGURE 5.3: lof_M^t and lof_I^t cases of insider threat instances. lof_M^t denotes the LOF for a minority instance A^t with respect to the k_{LOF} nearest neighbours from the majority class instances (M). lof_I^t denotes the LOF for a minority instance A^t with respect to the k_{LOF} nearest neighbours from the minority class instances ($I \setminus A^t$). The blue circles represent the majority instances and the red squares represent the minority instances.

The Percentile Rank for Minority Instances The inference from the values of lof_M^t and lof_I^t in each of the above mentioned cases controls the number of artificial samples to be generated per minority instance. In lof_M^t, we define $perclof_M^t$ as the percentile rank for each minority instance A^t compared to the majority instances. For instance, if the lof_M^t for A^t is the highest compared to all the majority instances (i.e. lof_M^t=1), then $perclof_M^t = 100$. If lof_M^t=0.7, then $perclof_M^t$=70; the lof_M^t for A^t is greater than 70% of the majority instances. Similarly, we define $perclof_I^t$ as the percentile rank for each minority instance A^t compared to other minority instances.

Removing Trapper Instances The anomalous behaviours attributed to malicious insiders often have a high resemblance with the normal behaviours of the insiders or their community (i.e. a group of users having the same role). In the feature space, this appears as minority (anomalous) instances located in a high-density region of majority (normal) instances. These minority instances are characterised by a low degree of outlierness (i.e. $low\ lof_M^t$ (close to 0)), thus a $low\ perclof_M^t$.

We define the parameter τ to be the survival threshold for the minority instances, such that each minority instance $A^t \in I$ having a $perclof_M^t < \tau$ is considered as a *trapper instance*. The high resemblance of the trapper instances with the surrounding majority instances, would *trap* the classifier to detect the surrounding majority instances as minority (FPs). In our AMOTRE oversampling technique, we propose to take a precautionary step and remove these trapper instances, in an attempt to reduce its contribution to flagging false alarms (FPs). Hence, a new set of minority instances I_r will include the minority instances having $percloft_M^t \geq \tau; \forall A^t$ excluding the removed *trapper instances* as detailed in Algorithm 1.

The idea of removing trapper instances is supported by the aim of the observation (1) in Section 5.1, where it is sufficient to detect any anomalous behaviour, not necessarily all behaviours, for each malicious insider threat. This means that removing a minority trapper instance from the set I is practically removing an anomalous behaviour from all the anomalous behaviours associated to a malicious insider threat. In other words, the *rest* of the anomalous behaviours associated with the malicious insider threat still exist, which permit the detection of the threat (i.e. insider) regardless of the removed anomalous behaviour. Therefore, removing trapper instances not only can improve the performance of the classifier, but also is supported by our ultimate aim to detect any-behaviour-all-threat.

Peculiarity of Remaining Minority Instances We define the peculiarity of a minority instance $A^t \in I_r$ using the probability p^t. p^t represents the probability of generating artificial samples for A^t. Consider the Eq. 5.9:

$$p^t = \left(\lceil perclof_M^t \rceil \times \lceil perclof_I^t \rceil\right) / 10^4 \tag{5.9}$$

where p^t is a probability devised from the product of $perclof_M^t$ and $perclof_I^t$. The rationale behind using the product is to utilise the degree of outlierness of the instance A^t with respect to both: (1) the majority instances ($perclof_M^t$), and (2) the minority instances ($perclof_I^t$). In other words, p^t actually determines the peculiarity of an instance A^t based on its location among the data distribution of both majority instances and minority instances. And accordingly, p^t gives the probability of generating artificial samples for A^t.

5.3.2.2 Generating Artificial Samples for Minority Instances

Let $perc.over$ represent the percentage of artificial samples to be generated, and let $numS = (perc.over/100) \times card(I_r)$ represent the number of artificial samples to be generated. In the following, we only consider the set of remaining minority instances I_r, excluding the removed trapper instances, to generate the artificial samples. As aforementioned, the peculiarity of the minority instances, which manifests in the probability p^t for each $A^t \in I_r$, will guide the process of sampling. The steps described below are repeated for a number of iterations until $numS$ of artificial samples is generated. Algorithm 2 gives a brief formalisation for generating artificial samples for minority instances.

Identifying the Chances for Sampling Minority Instances Per Iteration A probability distribution D_p is devised using the above probabilities $p^t \; \forall A^t \in I_r$ to determine whether an artificial sample S^t will be generated for A^t at the current iteration. We define $I_c \sim D_p$ where I_c represents the set of instances chosen according to D_p to be sampled at the current iteration, and p is a continuous range of the values of p^t. For instance, if $p^t = 1$, where $perclof^t_M = 100$ and $perclof^t_I = 100$, this means that A^t is an *extreme outlier* instance and it is safe to create a cloud of artificial samples around it as much as possible. However, as p^t decreases, the chance of generating artificial samples around A^t declines, because an *extreme outlier* instance attracts more artificial samples around it. For example, if $p^t = 0.25$, this may map to two cases: either $high\ lof^t_M$ (close to 1) and $low\ lof^t_I$ (close to 0), then A^t is surrounded by minority instances and we give a lower chance to generating artificial samples around A^t; or $low\ lof^t_M$ (close to 0), then A^t is surrounded by majority instances and we should seek to generate as few as possible artificial samples around it. In this way, the minority instances having higher peculiarity p^t are given more chance to generate artificial samples surrounding them.

Artificial Sampling At Feature Level Let $n^{t'}_f$ represent the value of the f^{th} feature of a majority instance $N^{t'} \in M$ at the session slot t', and let a^t_f represent the value of the f^{th} feature of a minority instance $A^t \in I_c$ at the session slot t. For each feature $f; 1 \preceq f \preceq m$, we calculate the distance from a^t_f of A^t to the nearest neighbour $n^{t'}_f$ of

Algorithm 2 Generating an artificial sample S^t

```
1: $numS \leftarrow (perc.over/100) \times card(I_r)$
2: while $card(I_s) < numS$ do
3:     $I_c \sim D_p$
4:     foreach $A^t \in I_c$ do
5:         $S^t \leftarrow$ GENERATESAMPLE($A^t$,$M$)
6:         $I_s \leftarrow I_s \cup S^t$
7:     end for
8: end while
9: return $I_s$
10:
11: function GENERATESAMPLE($A^t$,$M$)
12:     foreach $a_f^t$ $in$ $A^t$ do
13:         $pos\ n_f^{t'} \leftarrow$ positive nearest neighbour in $M$ at the level of feature $f$
14:         $neg\ n_f^{t'} \leftarrow$ negative nearest neighbour in $M$ at the level of feature $f$
15:         $dirS \sim D_{prob+}$
16:         if $\exists pos\ n_f^{t'} \wedge neg\ n_f^{t'}$ then
17:             $dirN \leftarrow dirS$
18:             if $dirS = +1$ then
19:                 $distN \leftarrow dist(a_f^t, pos\ n_f^{t'})$
20:             else
21:                 $distN \leftarrow dist(a_f^t, neg\ n_f^{t'})$
22:             end if
23:         end if
24:         if $\exists pos\ n_f^{t'}$ then
25:             $distN \leftarrow dist(a_f^t, pos\ n_f^{t'})$
26:         end if
27:         if $\exists neg\ n_f^{t'}$ then
28:             $distN \leftarrow dist(a_f^t, neg\ n_f^{t'})$
29:         end if
30:         $s_f^t \leftarrow a_f^t + dirS \times rand(0 : \lambda \times distN)$
31:         $S^t \leftarrow S^t \cup s_f^t$
32:     end for
33:     return $S^t$
34: end function
```

$N^{t'}$ at the level of feature f. In other words, we search the set of majority instances M at the level of feature f only, and we find the closest feature $n_f^{t'}$ for a_f^t. At the level of feature f, there exists two directions: positive ($+ve$), and negative ($-ve$). Thus, a_f^t may have (1) only a $+ve$ nearest neighbour, (2) only a $-ve$ nearest neighbour, or (3) both a $+ve$ nearest neighbour and a $-ve$ neatest neighbour. For instance, if a_f^t is greater than $n_f^{t'}$, then the nearest neighbour's distance $distN$=$dist(a_f^t, n_f^{t'})$ will be positive. On the other hand, if a_f^t is less than $n_f^{t'}$, then $distN$ will be negative. The value(s) of $distN$ for a_f^t is required to generate *artificial feature values* as demonstrated

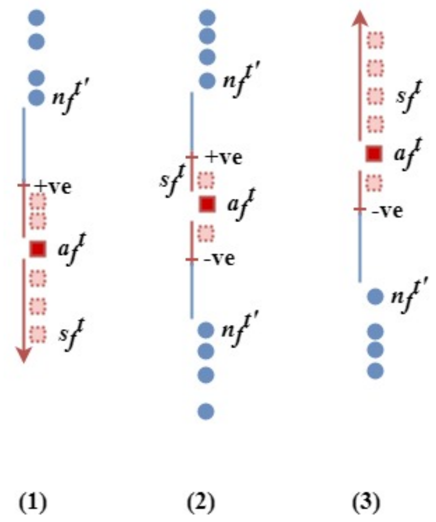


FIGURE 5.4: Cases of generating artificial insider threat samples in AMOTRE over one feature (one dimension). The blue circle represents a majority feature value $n_f^{t'}$, the red square represents a minority feature value a_f^t, and the red dashed square represents an artificial feature value s_f^t. Fig. 5.4(1) illustrates the case where a_f^t has only a $+ve$ nearest neighbour $n_f^{t'}$. Fig. 5.4(3) illustrates the case where a_f^t has only a $-ve$ nearest neighbour $n_f^{t'}$. Fig. 5.4(2) illustrates the case where a_f^t has two nearest neighbours in both directions.

in Equation 5.10.

We define $prob^+$ as the probability of generating an artificial sample in the positive ($+ve$) direction. A probability distribution D_{prob^+} is devised using the probability $prob^+$ to determine whether the direction $dirS$ of the artificial feature value s_f^t is $+1$ or -1. We define $dirS \sim D_{prob^+}$, where D_{prob^+} is a continuous range of the values of $prob^+$. Figure 5.4 illustrates generating an artificial feature value s_f^t for a_f^t at the level of feature f. Let a blue circle represent a majority feature value (i.e. feature value $n_f^{t'}$ of a majority instance), a red square represent a minority feature value (i.e. feature value a_f^t of a minority instance), and a red dashed square represent an artificial feature value (i.e. feature value s_f^t of an artificial instance) generated. In the following, we describe the three cases for generating an artificial feature value:

- If a_f^t has only a $+ve$ nearest neighbour $n_f^{t'}$ as Figure 5.4(1), then the probability of generating an artificial feature value s_f^t in the $+ve$ direction is lower. We set up $prob+ = 0.2$, such that the value of $prob^+$ is low (in the range $]0, 0.5[$) to give a less chance for generating artificial feature values in the $+ve$ direction. The rationale behind this is to give higher probability for sampling on the opposite direction of the $+ve$ nearest neighbour (i.e. $-ve$ direction), thus creating a cloud of artificial minority instances away from the nearest majority instances. If the chosen direction $dirS = +1$ based on D_{prob^+}, then s_f^t is calculated according to Eq. 5.10 such that $\lambda = 0.3$. Other wise, $\lambda = 1$.

- If a_f^t has only a $-ve$ nearest neighbour $n_f^{t'}$ as Figure 5.4(3), then the probability of generating an artificial feature value s_f^t in the $+ve$ direction is higher. We set up $prob^+ = 0.8$, such that the value of $prob^+$ is high (in the range $]0.5, 1[$) to give a more chance for generating artificial feature values in the $+ve$ direction. The rationale behind this is to give higher probability for sampling on the opposite direction of the $-ve$ nearest neighbour (i.e. $+ve$ direction), thus creating a cloud of artificial minority instances away from the nearest majority instances. If the chosen direction $dirS = +1$, then s_f^t is calculated according to Eq. 5.10 such that $\lambda = 1$. Otherwise, $\lambda = 0.3$.

- If a_f^t has two nearest neighbours in both directions, as in Figure 5.4(2), then the probability of generating an artificial feature value s_f^t in the $+ve$ and $-ve$ direction is equal. We set up $prob^+ = 0.5$, such that the value of $prob^+$ gives an equal chance ogenerating an artificial feature value both directions have equal chance of generating an artificial feature value. If the chosen direction $dirS = +1$ or $dirS = -1$, s_f^t is calculated according to Eq. 5.10 such that $\lambda = 0.3$.

$$s_f^t = a_f^t + dirS \times rand(0 : \lambda \times distN); s_f^t \geq 0 \tag{5.10}$$

where λ represents the parameter that controls the distance permitted to generate artificial feature values along the segment joining a_f^t and $n_f^{t'}$; and $rand(0 : \lambda \times distN)$ represents a random number generated to specify the location of the artificial sample along the segment joining a_f^t and $n_f^{t'}$ projected on the feature f. λ is set up to the value of 0.3 in the range $]0, 0.5[$, when the direction chosen to generate an artificial feature value is the same as the direction of the nearest neighbour. The rationale behind this is to locate the artificial feature values on the segment joining a_f^t and $n_f^{t'}$ away from the majority nearest neighbour $n_f^{t'}$, thus shielding the majority instances.

Otherwise, $\lambda = 1$, such that an artificial feature value can be located along the whole segment in the opposite direction.

Recall that all the artificial feature values in the generated community behaviour data sets are normalised to the range $[0, 1]$. Hence, if the calculated $s_f^t < 0$, we assign $s_f^t = min_f$ to avoid negative feature values. min_f represents the minimum value of the f^{th} feature among the minority instances.

Artificial Sampling At Instance Level Recall that if a minority instance A^t is given the chance to be sampled at an iteration (based on its peculiarity), then an artificial sample (instance) S^t is generated with a sampling process at the feature level. In other words, for each feature $f; 1 \preceq f \preceq m$, an artificial feature value s_f^t is generated. Accordingly, an artificial sample (instance) S^t=$\{s_1^t, s_2^t, ..., s_m^t\}; 1 \preceq f \preceq m$ associated with the minority instance A^t is generated.

As previously mentioned, the steps of Section 5.3.2.2 are repeated for a number of iterations until $numS$ of artificial samples are generated. Recall that I_r represents the set of remaining minority instances after removing *trapper instances*. Let I_s represent the set of artificial samples (instances) generated for I_r using the AMOTRE technique. At each iteration, the generated S^t is appended to I_s (I_s=$I_s \cup S^t$ as formalised in Algorithm 2). Consequently, the original set of minority instances I is updated to I=$I_r \cup I_s$ to comprise the set of remaining minority instances I_r (excluding *trapper instances*) and their generated minority samples I_s.

Figure 5.5 demonstrates the idea of generating artificial samples over two features (two dimensions). Let a blue circle represent a majority instance $N^{t'}$, a red square represent a minority instance A^t, and a red dashed square represent an artificial sample (instance) S^t. It is evident that the farther the minority instance A^t from the majority instances as well as other minority instances, the more chance is given to generate artificial samples around it. Furthermore, the artificial samples are mostly generated in the opposite direction of the nearest neighbours from the majority instances, thus shielding the border of the majority instances.

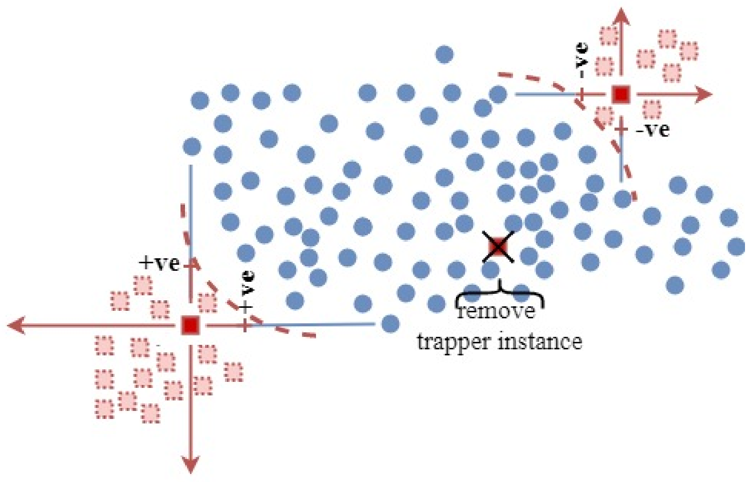


FIGURE 5.5: Cases of generating artificial insider threat samples in AMOTRE over two features (two dimensions). The blue circle represents a majority instance $N^{t'}$, the red square represents a minority instance A^t, and the red dashed square represents an artificial sample (instance) S^t. The red dashed arcs represent the shield for the border of the majority instances.

5.4 Experiments

In this section, we give a description of the variety of experiments carried out to evaluate the performance of the proposed approach, together with an explanation of the values used for parameter tuning. After that, we introduce the default versions and the refined versions of the evaluation measures utilised to evaluate the performance of the methods. Last but not least, we discuss the results of the experiments with a statistical significance test, and prove the merit of the proposed hybrid approach, CD-AMOTRE.

5.4.1 Experimental Setup

In order to evaluate the effectiveness of the proposed approach, we performed a variety of experiments on the CMU-CERT data sets on Windows Server 2016 on Microsoft Azure (RAM $140GB$, OS $64 - bits$, CPU Intel Xeon $E5 - 2673v3$). We implemented AMOTRE and carried out the experiments in R environment ($R - 3.4.1$) using **Rlof** package for LOF in AMOTRE, **DMwR** package for SMOTE, **caret** package [114] for the classification methods and their evaluation, and **MASS** package for the Wilcoxon Ranked test.

TABLE 5.1: Definition of Experiments.

SMOTE	SMOTE minority class I
CD(M)-SMOTE	SMOTE + CD of majority class M only
CD(MI)-SMOTE	SMOTE + CD of M and I
AMO-na	AMOTRE I without trapper removal
AMOTRE	AMOTRE I with trapper removal
CD(M)-AMOTRE	AMOTRE + CD of M only
CD(MI)-AMOTRE	AMOTRE + CD of M and I

5.4.2 Description of the Experiments

Table 5.1 presents the variety of experiments carried out to evaluate the performance of the proposed technique AMOTRE: without trapper removal; with trapper removal; with decomposition of the majority class M only; or with decomposition of the majority class M and the minority class I. These experiments are also performed over SMOTE to compare with AMOTRE using the measures defined later on: **F1 measure**, **TP**$_T$, **FP**. Note that the procedure of trapper removal is not applied with SMOTE, because it was first introduced as an essential part (-TRE part) in the proposed AMO-TRE technique. In the experiments, we analyse the influence of the -TRE part on our technique by applying the oversampling: (1) without trapper removal (AMO-na), and (2) with trapper removal (AMO-TRE). We then analyse the influence of introducing class decomposition variations to each of SMOTE and AMOTRE.

Each of the experiments is evaluated on five base classification methods: Random Forest (rf); Extreme Gradient Boosting (xgb); Support Vector Machines with linear kernel (svmL), polynomial kernel (svmP), and radial basis function kernel (svmR).

The experiments are tuned for different values of parameters as shown in Table 5.2. *Note that an extensive number of experiments was done to select the presented tuning values for the parameters. The values were selected based on the experiments achieving the best performance in terms of the evaluation measures described below.*

Regarding class decomposition, we tuned the number of clusters for k=$\{2, 4, 6\}$ for both (1) the decomposition of the majority class M, and (2) the decomposition of the minority class I. However, the results for only k=2 are reported, due to revealing better performance. The proposed approach was able to detect most of the malicious

TABLE 5.2: Tuned Parameters for SMOTE and AMOTRE.

k=$\{2, 4, 6\}$	number of clusters
$perc.over$=$\{200, 300, 400\}$	percentage of oversampling
τ=10	survival threshold for A^t in -TRE part
$prob^+$=$\{0.2, 0.5, 0.8\}$	controlled by the direction of $n_f^{t'}$
λ=$\{0.3, 1\}$	controlled by $dirS$ and $prob^+$

insider threats in the data sets for k=2. The literature demonstrated the effectiveness of k-means clustering for small values of k when applied for class decomposition [104], [109].

As explained in Section 4.3.2, a malicious insider threat comprises a complex pattern of anomalous behaviours carried out by a malicious insider. For instance, in community com-P, we have 17 malicious insider threats. These malicious insider threats are associated to 366 anomalous instances (behaviours) that make up the minority class I. Thus, when applying the class decomposition of the minority class I, we are actually clustering the anomalous training instances from the 366 instances appended to the artificial set I_s; not from the 17 malicious insider threats.

Regarding SMOTE technique, the oversampling percentage is tuned for $perc.over$ =$\{200, 300, 400\}$ to assess whether increasing the percentage of generated artificial samples to the minority class I improves the performance of SMOTE. The tuned values of $perc.over$ are selected to achieve an approximate balance between the sub-classes (clusters). Similarly, our AMOTRE technique is tuned for $perc.over$=$\{200, 300, 400\}$ to generate an equal number of artificial samples to that generated in SMOTE, in order to ensure a fair comparison between SMOTE and AMOTRE. Note that we report the results for only $perc.over$=200, due to achieving the lowest number of false positives (FP); which is the ultimate aim of our work.

The -TRE part is tuned for only a small value τ=10 to test the influence of removing trapper instances from the minority class instances on the overall performance of AMOTRE. We select the survival threshold τ=10 so that each minority instance $A^t \in I$ having a percentile rank $perclof_M^t < \tau$ is considered as a trapper instance and removed (as detailed in Section 5.3.2).

The details about tuning $prob^+$ and λ, for the displayed values in Table 5.2, can be found in the description of AMOTRE technique in Section 5.3.

For evaluation, we performed the experiments with 2-fold cross validation and 10-fold cross validation on each of the communities respectively. The 10-fold cross validation allows the base classification methods to learn on a 90% subsample of the data, and test on a 10% subsample repetitively for 10 times. This would best provide a comprehensive analysis of the performance of each experiment. However, the complexity of the insider threat problem is reflected in the scarcity of anomalous instances (minority instances) available in the CMU-CERT data. In the 10-fold cross validation, a 10% subsample testing data consists of a limited number of anomalous instances (behaviours), as well as a small number of normal instances (behaviours). Given this, it was challenging to show the merit of our approach compared to SMOTE experiments in terms of the number of FPs and the number of detected malicious insider threats. A 2-fold cross validation is also applied so that the base classification methods learn on a 50% subsample of the data, and test on a 50% subsample repetitively for 2 times. This allows the 50% subsample testing data to have more instances, including more anomalous instances (behaviours). Hence, it helps us to reveal the superiority of the performance of AMOTRE experiments compared to SMOTE experiments.

Whether in 2-fold cross validation or in 10-fold cross validation, each fold may contain a subset of the anomalous behaviours belonging to a malicious insider threat. In terms of the training, this provides 'weak supervision', as some of the anomalous behaviours associated to a particular threat will be missing in the training fold(s). In terms of the testing, this would show the 'robustness' of the approach being able to detect the malicious insider threat whose behaviours are partially represented in the training fold(s), even with a weak signal (threats are partially represented in the test fold(s)). The results for both 2-fold cross validation and 10-fold cross validation are reported in this Chapter.

We can sum up that 70 experiments are presented per community in terms of 7 types of experiments, 5 base classification methods, and 2 vs 10 fold cross validation. Regarding the tuning of parameters k and $perc.over$, we carried out 420 experiments per community.

5.4.3 Evaluation Measures

Much research has been done to detect or mitigate malicious insider threats, but standard measures have not been established to evaluate the proposed models [115]. The research practices show that the insider threat problem demands the measurement of the effectiveness of the models before being deployed, preferably in terms of true positives (TP) and false positives (FP) [116].

In the state-of-the-art, a remarkable number of approaches were validated in terms of FP measure [21], [42], [44], [58], [63]. This sheds light on the importance of the FP measure to address the shortcoming of the high number of false alarms (FPs). Furthermore, some approaches were validated in terms of: TP measure [42]; F1 measure [38], [44]; AUC measure [37], [51], [59]; precision and recall [33], [48]; accuracy [48], [63]; and others.

The variety of the utilised evaluation measures in the state-of-the-art reveals the critical need to formulate the insider threat problem and to define the measures that would best validate the effectiveness of the proposed approach. In the following, we define the evaluation measures utilised in this Chapter.

As previously mentioned, the ultimate aim of this approach is to detect any-behaviour-all-threats (*threat hunting* as described in Section 5.1 observation (1)), and to reduce the number of false alarms (based on observation (2)). In the following, we define the measures used to evaluate the performance. We introduced refined versions of some measures. The rationale behind this is related to our ultimate aim, which is to detect the malicious insider threats (not necessarily all anomalous behaviours per threat), and at the same time to reduce FPs (all false alarms of false predicted behaviours).

- P: *Positives* number of anomalous instances (anomalous behaviours);
- P_T: *Threats* number of malicious insider threats associated to anomalous instances. In other words, P_T is the number of malicious insiders attributed to the anomalous behaviours;
- **TP**$_T$: *True Positives* a refined version of the default TP to evaluate the number of threats detected by the system among all the P_T malicious insider threats.

TP$_T$ is incremented if at least one anomalous instance (behaviour associated to the threat) is predicted as anomalous;

- **FP**: *False Positives* number of normal instances (behaviours) that are detected as anomalous instances;
- TN: *True Negatives* number of normal instances (behaviours) that are predicted as normal;
- FN$_T$: *False Negatives* a refined version of the default FN to evaluate the number of insider threats not detected; and
- **F1** measure: defined based on the values of the above defined measures. Note that F1 is not close to 1 due to the use of refined versions of some measures, but this does not reflect low performance. However, knowing that the maximum TP$_T$ (evaluated per threat) is much lower than the minimum FP (evaluated per behaviour), the defined F1 measure closer to 0.5 reveals *good* performance.

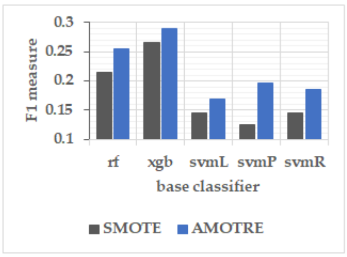
5.4.4 Results and Discussion

In this Section, we discuss the results of the variety of experiments carried out, and show the merit of the proposed approach CD-AMOTRE according to the following targets:

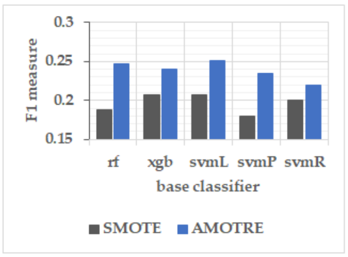
1. comparing SMOTE vs AMOTRE over the minority class;
2. assessing the influence of trapper removal on AMOTRE; and
3. assessing the effectiveness of class decomposition on the majority class and the minority class.

5.4.4.1 SMOTE vs AMOTRE over Minority Class

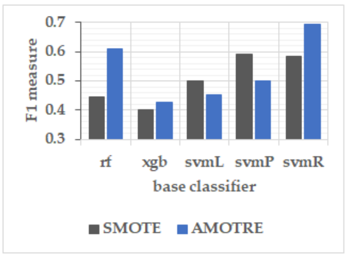
Figure 5.6 presents the value of F1 measure for each of SMOTE and AMOTRE on the minority class I over the communities in 2-fold cross validation. Figure 5.7 presents the value of F1 measure for each of SMOTE and AMOTRE on the minority class I over the communities in 10-fold cross validation. The results are reported with respect to the five base classifiers {rf, xgb, svmL, svmP, svmR}.



(A) com-P.

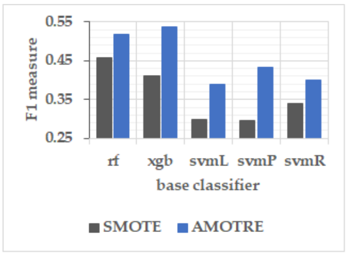


(B) com-S.

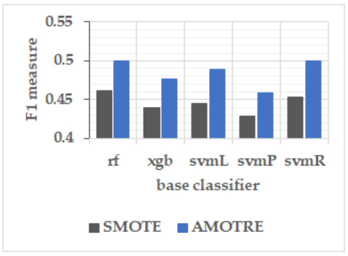


(C) com-I.

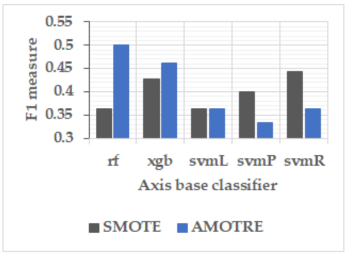
FIGURE 5.6: The value of F1 measure for SMOTE and AMOTRE over communities in 2-fold cross validation. The x-axis represents the five base classifiers {rf, xgb, svmL, svmP, svmR}, and the y-axis represents the values of F1 measure.



(A) com-P.



(B) com-S.



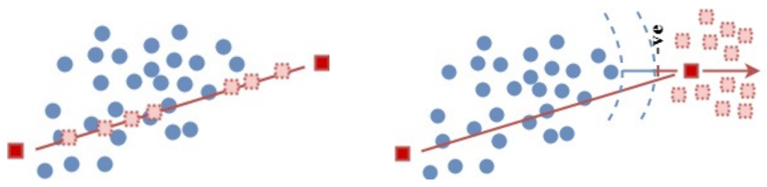
(C) com-I.

FIGURE 5.7: The value of F1 measure for SMOTE and AMOTRE over communities in 10-fold cross validation. The x-axis represents the five base classifiers {rf, xgb, svmL, svmP, svmR}, and the y-axis represents the values of F1 measure.

In 2-fold cross validation, the results show that AMOTRE outperforms SMOTE for all classifiers over com-P and com-S. Over com-P, AMOTRE achieves the maximum F1=0.2894;xgb compared to a maximum F1=0.2653;xgb for SMOTE. Over com-S, AMOTRE also achieves the maximum F1=0.2517;svmL compared to a maximum F1=0.2077;xgb for SMOTE. However, over com-I, AMOTRE shows better performance than SMOTE for all experiments except svmL and svmP. AMOTRE achieves the maximum F1=0.6956;svmR compared to the maximum F1=0.5925;svmP for SMOTE.

Similarly, in 10-fold cross validation, AMOTRE demonstrates a better performance compared to SMOTE for all classifiers over com-P and com-S, and for all classifiers except svmP and svmR over com-I.

Figure 5.8 demonstrates the advantage of AMOTRE over SMOTE according to each technique. Let the blue circles represent the majority instances, the solid-line red square represent the minority instances, and the red dashed squares represent the artificial samples generated to the minority class I. As shown in Figure 5.8a,



(A) SMOTE. (B) AMOTRE.

FIGURE 5.8: SMOTE vs AMOTRE in generating artificial insider threat samples. The blue filled circles represent the majority instances, the red filled squares represent the minority instances, and the red dashed squares represent the artificial samples of the minority instances. In Fig. 5.8a, SMOTE generates artificial samples along the segment line joining minority instances. In Fig. 5.8b, AMOTRE generates a cloud of artificial samples around the minority instance, and shields the majority instances.

SMOTE generates the artificial samples along the segment line joining minority instances, which results in minority artificial samples located among the majority instances. The high resemblance of the majority instances and the minority artificial samples contributes to the increase of FPs.

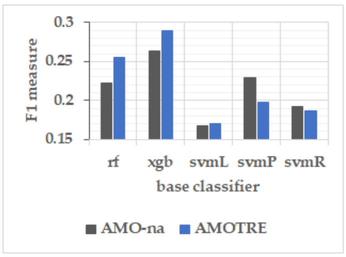
However, as shown in Figure 5.8b, AMOTRE first finds the nearest neighbour, for the minority instance, from the majority instances. It then generates a cloud of artificial samples around the minority instance, and shields the majority instances according to its predefined parameter λ. In this way, AMOTRE avoids generating minority artificial samples among majority instances, so that it reduces the number of FPs.

From the aforementioned analysis, we can conclude that the higher performance of AMOTRE compared to SMOTE is related to the merit of the proposed AMOTRE technique in reducing the number of FPs (target (1)).

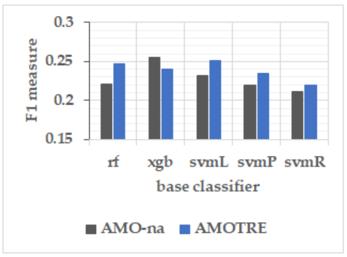
5.4.4.2 Influence of Trapper Removal on AMOTRE

The proposed technique AMOTRE consists of two key parts: the oversampling part (AMO-), and the trapper removal part (-TRE). In the following, we analyse the influence of the -TRE part on the AMOTRE technique.

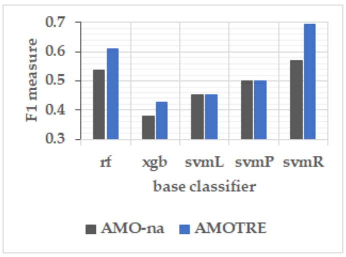
Figure 5.9 presents the value of F1 measure for AMO-na (AMO- without trapper removal) and AMOTRE (AMO-TRE with trapper removal) on the minority class I over the communities in 2-fold cross validation. Figure 5.10 presents the value of F1 measure for AMO-na and AMOTRE on the minority class I over the communities



(A) com-P.

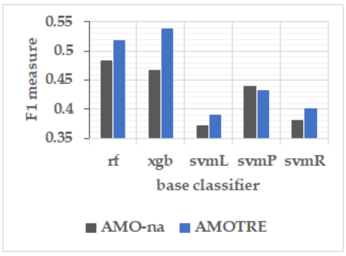


(B) com-S.

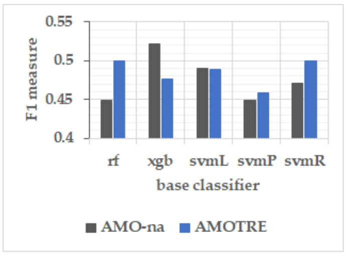


(C) com-I.

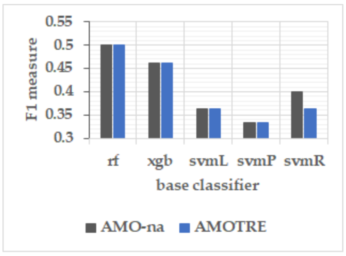
FIGURE 5.9: The value of F1 measure for AMO-na and AMOTRE over communities in 2-fold cross validation. The x-axis represents the five base classifiers {rf, xgb, svmL, svmP, svmR}, and the y-axis represents the values of F1 measure.



(A) com-P.



(B) com-S.



(C) com-I.

FIGURE 5.10: The value of F1 measure for AMO-na and AMOTRE over communities in 10-fold cross validation. The x-axis represents the five base classifiers {rf, xgb, svmL, svmP, svmR}, and the y-axis represents the values of F1 measure.

in 10-fold cross validation. The results are reported with respect to the five base classifiers.

In 2-fold cross validation, over com-P, AMOTRE reports a higher F1 measure than AMO-na for all classifiers except svmP and svmR. AMOTRE attains the maximum F1=0.2894;xgb compared to a maximum F1=0.2637;xgb for AMO-na. Over com-S, AMOTRE also reports a higher F1 than AMO-na for all classifiers except xgb. AMOTRE attains a maximum F1=0.2517;svmL compared to the maximum F1=0.2556;xgb for AMO-na. Over com-I, AMOTRE reports a higher F1 than AMO-na for all classifiers. AMOTRE attains the maximum F1=0.6956;svmR compared to a maximum F1=0.5714;svmR for AMO-na. Note that AMO-na shows equal performance AMOTRE in terms of F1 with respect to {svmL,svmP}.

Likewise in 10-fold cross validation, AMOTRE shows a better performance in terms of F1 measure compared to AMO-na with respect to most of the classifiers over all communities.

We can conclude that removing *trapper instances* from the minority class I boosts the performance of the base classifier in terms of F1 measure (target (2)). These trapper instances, if not removed, would be selected in the AMOTRE iterations to generate artificial samples around them. This would trap the classifier from finding the optimal decision boundary that separates majority instances from minority instances, which in turn results in a high number of false alarms. However, removing trapper instances is a precautionary step towards reducing the number of FPs, and achieving a higher F1 measure.

5.4.4.3 Effectiveness of Class Decomposition for Majority and Minority Class

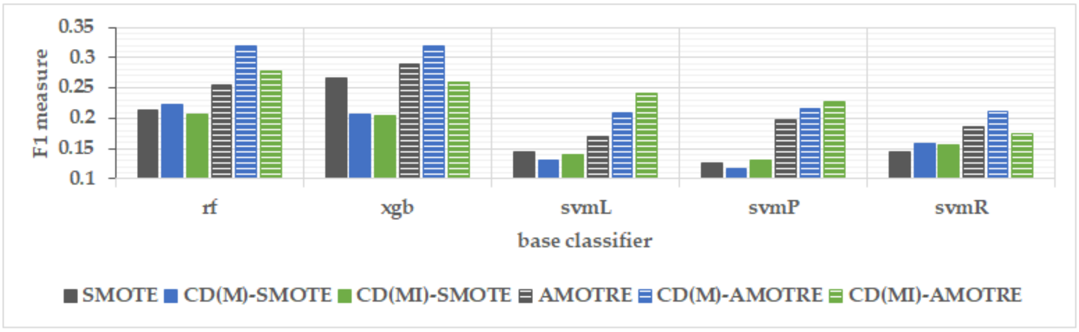
Figure 5.11 presents the value of F1 measure for different class decomposition approaches with SMOTE and AMOTRE over the communities in 2-fold cross validation. Figure 5.12 presents the value of F1 measure for different class decomposition approaches with SMOTE and AMOTRE over the communities in 10-fold cross validation. The results are reported with respect to the five base classifiers.

The first target is to analyse the effect of introducing class decomposition to each of the oversampling techniques SMOTE and AMOTRE. Class decomposition is applied in two different strategies: to the majority class M only as in CD(M)-SMOTE and CD(M)-AMOTRE; and to the majority class M as well as to the minority class I after applying the oversampling techniques as in CD(MI)-SMOTE and CD(MI)-AMOTRE. The second target is to compare the performance of AMOTRE alone to AMOTRE with class decomposition.

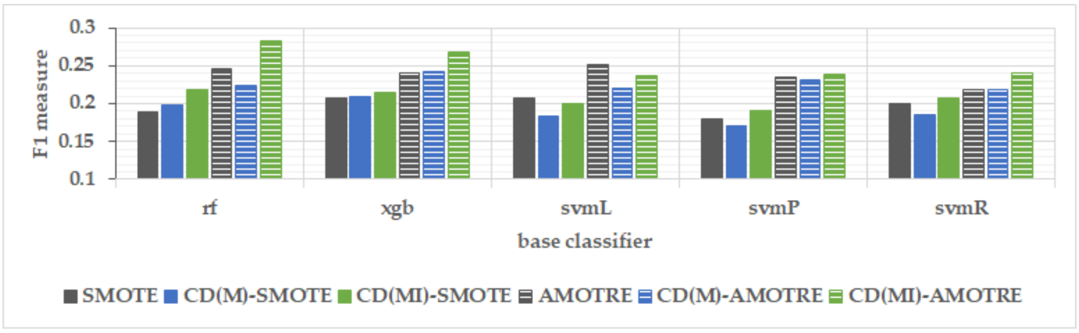
In 2-fold cross validation over com-P, CD(M)-AMOTRE outperforms all other SMOTE and AMOTRE experiments in terms of F1 with respect to {rf,xgb,svmR}. Whereas, CD(MI)-AMOTRE outperforms all experiments with respect to {svmL, svmP}. Overall, CD(M)-AMOTRE attains the maximum F1=0.32;{rf,xgb} followed with a maximum F1=0.2894;xgb for AMOTRE without class decomposition.

Over com-S, CD(MI)-AMOTRE outperforms all other SMOTE and AMOTRE experiments for all classifiers except svmL, where AMOTRE reports the best performance in terms of F1. Overall, CD(MI)-AMOTRE attains the maximum F1=0.2834;rf followed with a maximum F1=0.2517;svmL for AMOTRE.

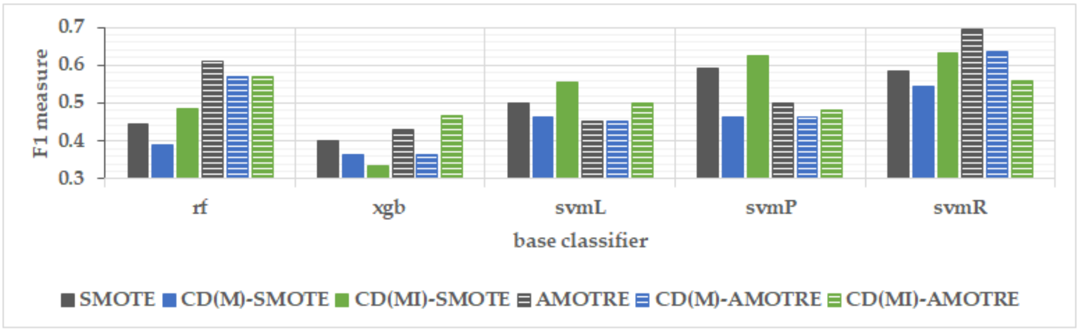
Over com-I, AMOTRE outperforms all other SMOTE and AMOTRE experiments with respect to {rf,svmR}. CD(MI)-AMOTRE outperforms all other experiments



(A) com-P.



(B) com-S.

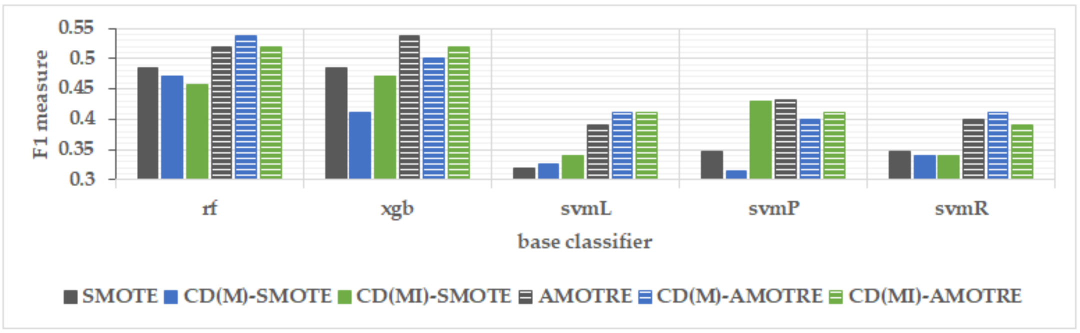


(C) com-I.

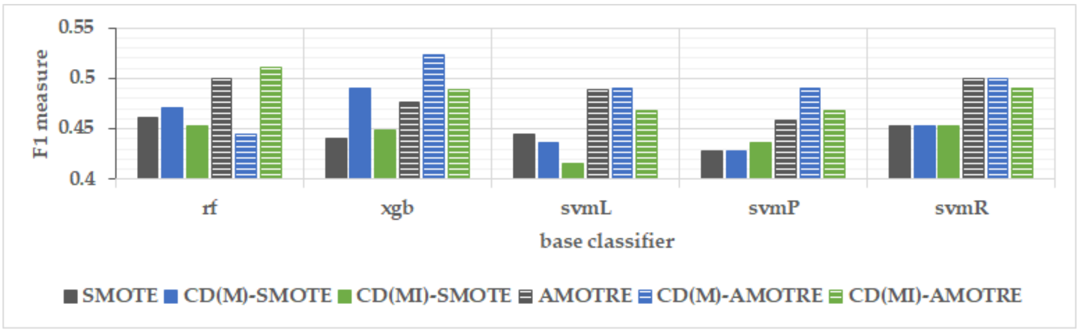
FIGURE 5.11: The value of F1 measure for SMOTE and AMOTRE with and without the class decomposition variations over communities in 2-fold cross validation. The x-axis represents the five base classifiers {rf, xgb, svmL, svmP, svmR}, and the y-axis represents the values of F1 measure.

with respect to xgb. However, CD(MI)-SMOTE wins all other SMOTE and AMOTRE experiments with respect to {svmL,svmP}. Overall, AMOTRE attains the maximum F1=0.6956;svmR followed with a maximum F1=0.6363;svmR for CD(M)-AMOTRE.

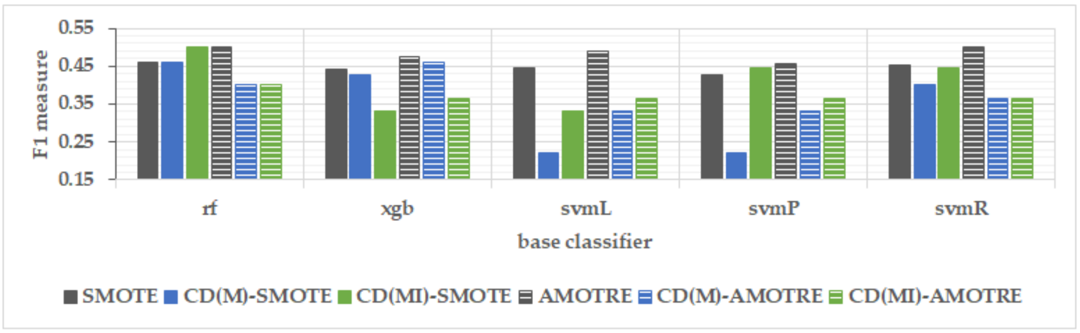
In 10-fold cross validation over com-P, CD(M)-AMOTRE shows the best performance compared to all other SMOTE and AMOTRE experiments with respect to {rf,svmL,svmR}. It is worth noting that CD(MI)-AMOTRE shows equal performance to CD(M)-AMOTRE in terms of F1 measure with respect to svmL. On the other hand, AMOTRE shows the best performance with respect to {xgb,svmP}.



(A) com-P.



(B) com-S.



(C) com-I.

FIGURE 5.12: The value of F1 measure for SMOTE and AMOTRE with and without the class decomposition variations over communities in 10-fold cross validation. The x-axis represents the five base classifiers {rf, xgb, svmL, svmP, svmR}, and the y-axis represents the values of F1 measure.

Over com-S, CD(M)-AMOTRE shows the best performance for all classifiers except rf, where CD(MI)-AMOTRE outperforms all other SMOTE and AMOTRE experiments. Note that AMOTRE shows equal performance to CD(M)-AMOTRE in terms of F1 measure with respect to svmR.

Over com-I, AMOTRE shows the best performance for all classifiers, though CD(MI)-SMOTE shows equal performance to AMOTRE in terms of F1 measure with respect to rf.

We can deduce that introducing class decomposition to the oversampling technique AMOTRE revealed better performance compared to AMOTRE with no class decomposition, as well as all SMOTE experiments (target (3)). It is quite remarkable that each of CD(M)-AMOTRE and CD(MI)-AMOTRE reported the best performance in terms of achieving the highest F1 measure alternatively among the communities in 2-fold and 10-fold cross validations.

Statistical Significance: Wilcoxon Signed-Rank Test To test the significance of the results, we use the Wilcoxon Signed-Rank Test which compares each pair of experiments. All SMOTE experiments are compared against all AMOTRE experiments in terms of F1 measure. Table 5.3 tabulates the p-value calculated for each pair of experiments with respect to all base classifiers in 2-fold and 10-fold cross validations at $.05$ significance level. For instance, the p-value for $\langle$SMOTE, AMOTRE$\rangle$ pair in 2-fold cross validation (first row) is a result of 'the values of F1 measure for SMOTE for all base classifiers $\{$rf, xgb, svmL, svmP, svmR$\}$' compared with 'the values of F1 measure for AMOTRE for all base classifiers.'

Table 5.3 shows that SMOTE (without class decomposition) is significantly different from AMOTRE reporting a p-value=$3.015e-02 < .05$ in 2-fold cross validation and a p-value=$3.83e-02 < .05$ in 10-fold cross validation. It also shows that CD(M)-SMOTE is significantly different from CD(M)-AMOTRE reporting a p-value=$2.105e-03 < .05$ in 2-fold cross validation and a p-value=$8.253e-03 < .05$ in 10-fold cross validation. Regarding CD(MI)-SMOTE and CD(MI)-AMOTRE, the p-values do not indicate a significant difference for this pair of experiments, where a p-value=$9.46e-02 > .05$ is reported in 2-fold cross validation and a p-value=$3.485e-01 > .05$ is reported in 10-fold cross validation.

Evaluation of TP$_T$ and FP Measures. The following will address the TP$_T$ and FP measures, which represent key measures for the insider threat problem. Table 5.4 reports the maximum TP$_T$ in each of the SMOTE and AMOTRE experiments over the communities, associated with the base classifier(s) which achieved the maximum TP$_T$. In Table 5.5, we report the minimum FP measure in each experiment over the communities associated with the base classifier(s) which achieved the minimum FP.

TABLE 5.3: Wilcoxon Signed-Rank Test at .05 significance level. The p-value is calculated for each pair of experiments in 2-fold cross validation and in 10-fold cross validation. For instance, the p-value for ⟨SMOTE, AMOTRE⟩ pair in 2-fold cross validation (first row) is a result of 'the values of F1 measure for SMOTE for all base classifiers {rf, xgb, svmL, svmP, svmR}' compared with 'the values of F1 measure for AMOTRE for all base classifiers.'

CV folds	Experiments pair	p-value
2	SMOTE, AMOTRE	3.015e-02
2	CD(M)-SMOTE, CD(M)-AMOTRE	2.105e-03
2	CD(MI)-SMOTE, CD(MI)-AMOTRE	9.46e-02
10	SMOTE, AMOTRE	3.83e-02
10	CD(M)-SMOTE, CD(M)-AMOTRE	8.253e-03
10	CD(MI)-SMOTE, CD(MI)-AMOTRE	3.485e-01

TABLE 5.4: Maximum TP_T/P_T of detected insider threats over communities associated with the base classifier(s) which achieved the maximum TP_T. Note that the parameter combination(s) which achieved the maximum TP_T are listed.

CV folds	Experiment	com-P	com-S	com-I
2	SMOTE	13/16 {xgb,svmL}	17/21 svmR	**9/12** xgb
	CD(M)-SMOTE	**14/16** svmR	17/22 {xgb,svmL}	8/12 {rf,xgb}
	CD(MI)-SMOTE	13/16 svmL	**17/20** {xgb,svmP}	8/12 rf
	AMOTRE	13/16 {svmL,svmP}	**18/21** svmL	8/12 {svmP,svmR}
	CD(M)-AMOTRE	**14/16** {svmL,svmR}	17/22 ∀\ svmR	8/12 rf
	CD(MI)-AMOTRE	13/16 {svmL,svmP}	**18/21** {rf,xgb,svmL}	8/12 {rf,svmL}
10	SMOTE	8/10 {rf,xgb,svmP}	12/15 rf,svmP,svmR	3/7 ∀\rf
	CD(M)-SMOTE	8/10 ∀\xgb	12/15 $forall$	3/7 {rf,xgb}
	CD(MI)-SMOTE	8/11 ∀	12/15 {svmP,svmR}	**3/6** rf
	AMOTRE	8/10 svmP	12/15 svmR	3/7 ∀ \ {svmP,svmR}
	CD(M)-AMOTRE	8/10 svm-	12/15 svm-	3/7 xgb
	CD(MI)-AMOTRE	8/11 svm-	12/15 {rf,svmR}	2/6 ∀ \ {rf,svmR}

∀ | for all base classifiers
∀\xxxx | for all base classifiers except xxxx
svm- | for all utilised SVM methods

In 2-fold cross validation over com-P, it shows that CD(M)-SMOTE and CD(M)-AMOTRE achieve the maximum TP_T=14/16, where only 2 malicious insider threats are not detected. Note that 14/16 represents the number of detected threats TP_T=14 out of the number of threats P_T=16. Furthermore, CD(MI)-AMOTRE reduces the number of FPs to the minimum FP=46;rf (3.36%), compared to a minimum FP=69;xgb for SMOTE.

Over com-S, CD(MI)-SMOTE, AMOTRE and CD(MI)-AMOTRE attain the maximum TP_T, where CD(MI)-SMOTE detects TP_T=17/20; while AMOTRE and CD(MI)-AMOTRE detect TP_T=18/21. Only 3 malicious insider threats are missed. Furthermore, AMOTRE and CD(MI)-AMOTRE attain the minimum FP=88;xgb, compared

TABLE 5.5: Minimum FP over communities associated with the base classifier(s) which achieved the minimum FP. Note that the parameter combination(s) which achieved the minimum FP are listed.

cv Folds	Experiment	com-P	com-S	com-I
2	SMOTE	69 xgb	117 xgb	5 svmR
	CD(M)-SMOTE	80 xgb	123 rf	4 svmR
	CD(MI)-SMOTE	80 rf	118 rf	**0** svmP
	AMOTRE	49 xgb	**88** xgb	3 svmR
	CD(M)-AMOTRE	47 rf,xgb	101 xgb	3 svmR
	CD(MI)-AMOTRE	**46** rf	**88** xgb	6 svmR
10	SMOTE	15 {rf,xgb}	23 xgb	**1** svmR
	CD(M)-SMOTE	16 rf	22 xgb	**1** $\forall \setminus$ {rf,xgb}
	CD(MI)-SMOTE	16 rf	23 xgb	**1** {svmP,svmR}
	AMOTRE	**9** xgb	19 xgb	2 {rf,svmR}
	CD(M)-AMOTRE	**9** rf	**16** xgb	**1** rf
	CD(MI)-AMOTRE	**9** {rf,xgb}	18 xgb	**1** rf

to a minimum FP=117;xgb for SMOTE.

Over com-I, SMOTE attains the maximum TP_T=9/12, where only 3 malicious insider threats are missed. Furthermore, CD(MI)-SMOTE attains the minimum FP=0; svmP, compared to a minimum FP=3;svmR for AMOTRE and CD(M)-AMOTRE.

In 10-fold cross validation, as aforementioned, given a 10% subsample testing including a limited number of instances, it is challenging to show the merit of our approach compared to SMOTE experiments in terms of detected threats and reducing the number of FPs. Unlike 2-fold cross validation, it is evident that TP_T in 10-fold cross validation shows no significant difference among the SMOTE and AMOTRE experiments over all communities. Regarding the number of FPs, over com-P, all AMOTRE experiments (without and with class decomposition) attain the minimum FP=9 compared to a minimum FP=15 for SMOTE with respect to rf and xgb. Over com-S, CD(M)-AMOTRE attains the minimum FP=16 compared to FP=22 for CD-(M)SMOTE with respect to xgb. Over com-I, all SMOTE and AMOTRE experiments attain the minimum FP=1 except for AMOTRE without class decomposition which attains a minimum FP=2 with respect to {rf,svmR}.

In conclusion, AMOTRE experiments with class decomposition, namely CD(M)-AMOTRE and CD(MI)-AMOTRE, demonstrate the best performance in terms of TP_T and FP measures in 2-fold and 10-fold cross validations over com-P and com-S. It is worth noting that the number of FPs is reduced remarkably over AMOTRE experiments compared to SMOTE experiments, especially in 2-fold cross validation.

On the other hand, over com-I, it is quite evident that SMOTE experiments (with and without class decomposition) outperform AMOTRE experiments in terms of TP_T and FP measures in 2-fold and 10-fold cross validations. In Table 5.4, SMOTE and CD(MI)-SMOTE reveal the capability to detect one more malicious insider threat compared to other experiments in 2-fold and 10-fold cross validation respectively. In Table 5.5, CD(MI)-SMOTE reveals the capability to attain 0 FPs, compared to a minimum FP=3 attained by AMOTRE and CD(M)-AMOTRE in 2-fold cross validation. Note that CD(MI)-SMOTE is still based on the proposed class decomposition, however, not using AMOTRE technique, but SMOTE. Unlike 2-fold, all SMOTE and AMOTRE experiments in 10-fold experiments, excluding AMOTRE without class decomposition, show competitive performance in terms of FP measure (all attained the minimum FP=1 excluding FP=2 for AMOTRE). We argue that the poorer performance of AMOTRE compared to SMOTE experiments in terms of TP_T and FP measures over com-I is correlated to the peculiarities of the scenarios included in com-I. *First*, in the utilised community data sets (com-P, com-S, and com-I), only com-I includes malicious insider threats which map to **scenario s3**.

Second, the community com-I includes 2 malicious inside threats (i.e. malicious insiders) of scenario s2, and 10 malicious insider threats of scenario s3. We notice that 50 anomalous behaviours represent the 10 malicious insider threats of scenario s3, which *is much less* than 82 anomalous behaviours representing *only* 2 malicious insider threats of scenario s2. *We argue that the poor representation of the malicious insider threats of scenario s3* causes the lower performance of AMOTRE experiments in com-I. The poor representation may relate to the peculiarity of the scenario s3.

Third, we recall that scenario s3 considers a masquerader who gains access to their supervisor's PC using a keylogger and sends an alarming mass email to employees in the organisation. A masquerade threat is simulated on two users: an unauthorised user (i.e. masquerader), and an authorised user (i.e. victim). So, the masquerade threat may not be detected in a user's behaviour model, but in a community behaviour model, where the behaviours of multiple users having the same role is represented. The idea of masquerade threat is addressed in this thesis by constructing the community behaviour profiles (as discussed in Section 4.3) to build the models later on. The community behaviour model captures the co-occurrence of the changes of users' behaviours in the community. Thus, the co-occurrence of the changes of masquerader's behaviour and victim's behaviour may be detected. However, the masquerader's attack is on the victim's PC using their credentials and authorised privileges. Hence, the detection system would be deceived by the masquerader's attack as authorised on the victim's PC. *We argue that the complexity of scenario s3* causes the low performance of AMOTRE experiments in com-I.

Therefore, the limitation related to scenario s3 will be addressed in future work as discussed in Chapter 8.

To sum up The above experiments proved that AMOTRE outperforms SMOTE achieving a higher F1 measure over the communities (target (1)). We showed that the *trapper removal* is a key part of AMO-TRE oversampling technique, where AMOTRE reveals higher F1 measure with respect to the base classifiers over all communities (target (2)). Moreover, introducing class decomposition to each of the oversampling techniques SMOTE and AMOTRE improved the performance of the base classifiers (target (3)). The results show the merit of our hybrid approach CD-AMOTRE that integrates the class decomposition concept and the proposed AMOTRE oversampling technique in terms of highest F1 measure, maximum number of detected threats TP_T, and minimum number of FPs.

5.5 Summary

In this Chapter, we address the insider threat problem as a class imbalance problem with the aim of detecting malicious insider threats while reducing the number of false alarms. We propose a hybrid approach, namely CD-AMOTRE, which combines the class decomposition and AMOTRE oversampling and trapper removal technique.

The aim of adopting class decomposition is to weaken the majority class by decomposing it into subclasses (clusters), so the decision of the classifier would not bias toward the majority class. Class decomposition can also be applied to the minority class after oversampling in order to achieve balance between the subclasses. Hence, the result of class decomposition is transforming the two-class classification with class imbalance into multi-class classification with balanced subclasses (clusters).

We define AMOTRE as a selective oversampling technique, where it selects the minority instances to be oversampled based on a measured local outlier factor. The minority instances located in a high-density region of majority instances are trapper instances that would trap the classifier. In AMOTRE, trapper instances are removed to reduce the number of false alarms. Furthermore, AMOTRE constrains the generation of artificial samples to the minority class using the parameter λ to shield the border of the majority class. It allows generation of more artificial samples for minority instances away from the majority class than the ones closer to it. Hence, the oversampling procedure together with trapper removal (-TRE part) contributed in reducing FPs.

We evaluate different variations of applying AMOTRE with and without class decomposition compared to the state-of-the-art sampling technique SMOTE with respect to five high performing base classifiers. First, the results show that AMOTRE outperforms SMOTE achieving a higher **F1 measure**. Second, introducing class decomposition shows better performance for both CD-SMOTE and CD-AMOTRE with class decomposition against SMOTE and AMOTRE, where the best maximum number of detected threats by both was **14/16** (87.5%). However, CD-AMOTRE outperforms CD-SMOTE in terms of FPs, where the best minimum FP reported by CD(MI)-AMOTRE **46** (3.36%) is much lower than the minimum **FP** reported by SMOTE **69**. It is worth noting that the class decomposition contributed into reducing the FPs, though the class data may not be perfectly decomposed. In conclusion, CD-AMOTRE provided better performance in terms of higher F1 measure and lower number of false alarms.

Chapter 6

Anomaly Detection for Insider Threat Detection

6.1 Introduction

Anomaly detection tackles the rare-class problem by building a model based only on the normal class label, then predicting whether acquired new data is normal or anomalous. This class of anomaly detection methods is referred to as semi-supervised anomaly detection in [117]. However, semi-supervised classification refers to those techniques that operate having only a small set of labelled instances along with a typically larger set of unlabelled ones, regardless of the assigned class [118]. Thus, in this chapter, we use the term anomaly detection to refer to the one-class machine learning problem.

The ongoing monitoring and logging of the insiders' activities establishes huge useful data sets of information to learn the normal baseline of behaviour. However, the absence of previously logged malicious insider threats among the logs of normal users' behaviour in an organisation (the case of medium data maturity), shapes the insider threat detection mechanism into a one-class data mining approach, namely anomaly detection.

The 'behaviour-based' anomaly detection approaches reviewed in Section 2.3.2.2 have shown merit in addressing the insider threat detection problem, however, they do suffer from a high number of false alarms. For instance, a recent anomaly detection approach, called RADISH [21], reports a False Alarm rate=92% (refer to Table 2.2). This means that 92% of the alarms flagged are actually benign (FPs).

To tackle the shortcoming of the high number of FPs, in this Chapter we propose an anomaly detection framework that consists of two components: one-class modelling component, and progressive update component. First, the role of the one-class modelling component includes the decomposition of the normal class data into k clusters, and the training of an ensemble of k base anomaly detection methods (One-class Support Vector Machine –ocsvm– or isolation Forest –iForest–). Each base model of the ensemble is trained over one cluster, resulting in an ensemble of k base models. This allows the detection of anomalous *trapper* instances that have a high resemblance to normal instances, which would not be detected by the method if trained over the whole normal class data. Second, the role of the progressive update component is to adapt the decision boundary of each base model with sequentially acquired FP chunks. It includes a selective oversampling method to generate artificial samples for FPs per chunk, with the aim to reduce the number of FPs.

The proposed anomaly detection framework provides the following major contributions:

- a class decomposition method on the normal class data to detect the anomalous *trapper* instances;
- a progressive update method with sequentially acquired FP chunks to address the shortcoming of high number of FPs;
- an outlier-aware artificial oversampling method for FPs to avoid model overfitting by intelligently adapting decision boundaries of base models of the ensemble fed with the synthetically oversampled data; and
- a thorough performance evaluation and a statistical significance test of a variety of experiments utilising ocsvm and iForest, validating the effectiveness of class decomposition, progressive update, and oversampling, compared to that of base ocsvm and base iForest.

The rest of the Chapter is organised as follows. In Section 6.2, we give a background of the utilised anomaly detection methods (ocsvm and iForest). In Section 6.3, we propose an anomaly detection framework, with a detailed description of its components. In Section 6.4, we present the experimental setup and the refined versions of evaluation measures. We then evaluate a variety of experiments, to assess the effectiveness of the proposed framework and its components. Finally, we conclude this Chapter with a summary in Section 6.5.

6.2 Background

The approach to address the insider threat problem depends on the availability of data; whether the organisation historically collected system and network logs of the users' activities. If the data is available, it would either consist of normal instances, or normal instances with insider threat instances based on whether insider attacks previously occurred in the organisation. In this Chapter, we focus on the cases when the maturity of data availability in the organisation is medium (i.e. data logs for only normal behaviour is available). Based on this, the insider threat problem may be addressed from the perspective of anomaly detection.

The following gives a description of two popular anomaly detection methods: One Class Support Vector Machine (ocsvm) and isolation Forest (iForest), which will be used as base methods in the proposed framework to detect malicious insider threats. We adopted ocsvm and iForest, because each of the methods has been utilised for insider threat detection, either as a base method for the proposed approaches [22], [51], [63], or as a benchmark against which the performance of a deep learning approach was compared to its performance [61].

6.2.1 One Class Support Vector Machine

One Class Support Vector Machine (ocsvm) is a one-class classification method that learns from a target class and finds the hyperplane that separates the target class instances from the origin. ocsvm addresses the anomaly detection problem by building a model from normal class instances only. New data instances are then classified as normal or anomalous based on their deviation from the model. Kernels may be applied to map the data set into a high dimensional space and find a linear maximum-margin hyperplane that may be non-linear in the original feature space.

The availability of data logs for only normal behaviour makes ocsvm well suited to tackle the insider threat detection problem. ocsvm builds a model based on normal instances only, and then identifies malicious insider threats that deviate significantly from the pattern of normal behaviour.

6.2.2 Isolation Forest

isolation Forest (iForest) [50] is a model-based anomaly detection method that isolates anomalies from the normal instances instead of profiling normal instances. Let ψ represent a parameter that controls the size of the subsample extracted from the original data set to construct each isolation Tree (iTree). iTree is constructed by recursively partitioning data subsample instances until a specific tree height is attained. The tree height is derived from the predefined parameter subsampling size ψ. The task of anomaly detection requires finding an anomaly score for each instance, so that instances having a high degree of anomaly or short path lengths along the iTrees are more likely to be anomalous instances. iForest is defined as an ensemble of nt iTrees that provides the average path length for each instance over nt number of trees to accomplish convergence of results.

The application of iForest on the users' behaviour separates the malicious insider threats from the normal instances without constructing a behaviour profile.

6.3 Adaptive One-Class Ensemble-based Anomaly Detection

This Section presents the proposed anomaly detection framework for insider threat detection and provides a detailed description of its components. Figure 6.1 illustrates two phases: one-class modelling component and progressive update component.

6.3.1 One-class Modelling Component

The one-class modelling component acquires training data in the initial modelling phase. It consists of a clustering component that applies k-means clustering method on the normal class data in order to create k clusters. It then trains each cluster on

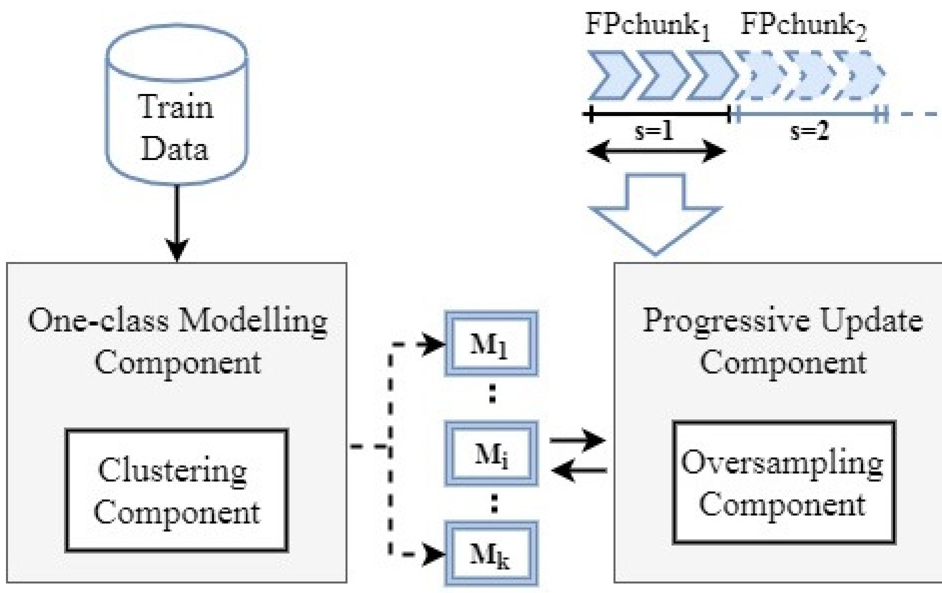


FIGURE 6.1: Proposed conceptual framework for anomaly detection. The framework has two components: a one-class modelling component which includes a clustering component, and a progressive update component which includes an oversampling component. $\{M_1, M_2, ..., M_k\}$ represent the set of models generated by the ensemble. The set of blue arrows represent the testing instances declared as FPs, and each segment of blue arrows represents an FP chunk acquired sequentially.

a base method. The result is an ensemble of a base method over k clusters, resulting in k models. In this work, we utilised two high-performing anomaly detection methods, ocsvm and iForest, as base methods in the proposed framework to detect malicious insider threats.

Clustering Component In Chapter 5, class decomposition was used to weaken the effect of the majority class in classification. However, in this chapter, we aim to cluster the normal class data to identify patterns in data, then build a one-class classifier on each cluster. In this way, the local anomalies can be identified, which is not the case if we used normal data as a whole.

Malicious insiders have authorised access to the network, system, and data, and are aware of the system management and security policies. These aspects aid the malicious insider to deceive the detection system, where some anomalous behaviour may have a high resemblance with the normal user's behaviour. This manifests as local anomalous instances located among normal instances. To address this issue, we apply class decomposition [77] on the normal class data. The idea is to decompose the normal class data into clusters and to train a detector per cluster, giving more opportunity for the detector to identify local anomalous instances with respect to a cluster that might not be detected over the whole data. We utilise k-means clustering method to identify patterns in the normal class data, given its efficiency. More information regarding the literature of class decomposition and the argument for selecting the k-means clustering method can be found in Section 5.3.1.

Let X^t=$\{x_1^t, x_2^t, ..., x_m^t\}$ represent the feature vector at session slot t, where $x_f^t; 1 \preceq f \preceq m$ represents the value of the f^{th} feature. Let y represent the normal class label. Each instance (i.e. feature vector) X^t either belongs or does not belong to normal class y. Let N=X^t $\forall t; X^t \in y$ represent the set of instances that belong to the normal class y. If we apply k-means clustering method on the set N, then N decomposes into k clusters. Let C=$\{C_1, C_2, ..., C_k\}$ represent the set of k clusters.

Fig. 6.2 represents the normal instances with blue circles, and the anomalous instances with squares. Let the solid-line outer circle represent the decision boundary generated by the base method over the whole normal data to separate the normal instances from the anomalous instances. Consider k=2, so that the normal data instances are grouped into 2 clusters. We construct an ensemble of k base methods and train a base method on each cluster of the k clusters. Let the inner dashed circles represent the decision boundaries generated by each cluster's base method. Fig. 6.2 reveals *two types of anomalous instances* defined below:

- *Borderline and outlier instances*: represented by red filled squares. Those instances are located at the borderline with respect to the solid-line outer decision boundary, or are far outliers; and

- *Trapper instances*: represented by red empty squares. Those instances are located in a sparse or dense area of normal instances.

The borderline and outlier anomalous instances can be easily detected by one of the aforementioned base methods trained over the whole normal data. However, as shown in Fig. 6.2, the solid-line outer decision boundary would not be able to detect trapper instances. As aforementioned, the trapper instances are located among

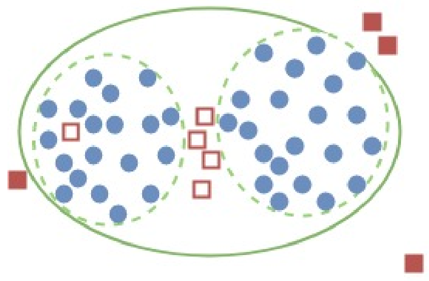


FIGURE 6.2: Clustering normal class instances. The blue circles represent the normal instances, and the squares represent the anomalous instances. The red filled squares represent borderline and outlier instances, and red empty squares trapper instances. The solid-line outer circle represents the decision boundary generated by the base method over the whole normal data. The inner dashed circles represent the decision boundaries generated by each cluster's base method.

normal instances, and therefore the base method would declare as normal instances (e.g. iForest would not assign a high anomaly score).

To address this issue, we propose to decompose the normal class data into clusters and to train an ensemble of a base method per cluster, so that the trapper instances can be identified by the base method(s) as anomalous with respect to cluster(s). Fig. 6.2 shows that the inner dashed decision boundaries, generated by clusters, can detect the trapper instances located in the disjoint area (i.e. sparse area of normal instances). Nevertheless, some trapper instances would still exist inside the clusters which may not be detected as positives, due to their existence in an inner dense area of normal instances.

Upon decomposing the normal training data set into k clusters, the role of one-class modelling component is to train an ensemble of a base method on the k clusters to generate k initial models. Let M=$\{M_1, M_2, ..., M_k\}$ represent the set of models generated by the ensemble. Each initial model M_i for $C_i; 1 \leq i \leq k$ is then used to detect the malicious insider threats in the testing data set. The decision d_i^t for a testing instance X^t with respect to M_i comes in Boolean form of {True, False}. In case of ocsvm, an instance X^t either belongs or does not belong to the normal class y. In case of iForest, X^t is identified as an anomalous instance if its anomaly score is greater than a defined anomaly score threshold τ. The parameter τ requires to be tuned, where an optimal isolation of anomalous instances is achieved at a certain threshold τ, as examined later in Section 6.4.

After that, the ensemble acquires a set of decisions D^t={$d_1^t, d_2^t, ..., d_k^t$} for a testing instance X^t to vote whether X^t is normal or anomalous. Each decision $d_i^t \in D^t; 1 \leq i \leq k$ is taken by a model M_i for a cluster C_i in the ensemble. The voting mechanism is executed as follows: (1) If $d_i^t \in D^t$ votes for anomalous $\forall i$ (i.e. by all models), then the overall decision D^t declares X^t as anomalous behaviour, and consequently flags an alarm warning of a malicious insider threat; (2) If $\exists d_i^t \in D^t$ votes for normal, then the overall decision D^t declares X^t as normal behaviour.

6.3.2 Progressive Update Component

The importance of the insider threat problem requires a continuous monitoring of the implemented detection system by the system's administrator(s). A false alarm is a result of a normal behaviour detected as anomalous by the system. This maps to normal instances that have a high similarity with anomalous instances, thus appear as suspicious events. To address this issue and to reduce the number of false alarms raised, we introduce the progressive update component. The role of this component is to progressively update the detection system with acquired FP chunks. We define an FP chunk as follows:

Definition 6.3.1: ***FP Chunk*** An FP chunk is a segment of capacity c that accumulates test instances declared as FPs. Let FPchunk$_s$={FP^1, FP^2, ..., FP^c} represent an FP chunk acquired at sequence s, such that D^t for each FP^t $\in$ FPchunk$_s$; $1 \leq t \leq c$ is predicted as a positive (i.e. anomalous instance) while the actual class label for FP^t is normal. Thus, each FP^t is declared as FP.

Consider, in Fig. 6.1, the set of blue arrows represent the testing instances declared as FPs, and each segment of blue arrows represents an FP chunk acquired sequentially. For example, the segment of blue arrows at sequence s=1 represents FPchunk$_1$. We define c as the capacity (size) of FP chunks; each FP chunk FPchunk$_s$ can accumulate c number of FPs. Fig. 6.1 illustrates FP chunks of capacity c=3; each FP chunk accumulates 3 FPs.

The progressive update method relies on the continuous monitoring by the administrator to investigate whether the flagged alarms are true or false. Along the run of the detection system, when an alarm is flagged, the administrator is required to investigate the suspected user and decide whether it is a malicious insider threat or a false alarm. In the latter case, the FP (false alarm) is accumulated into the FP chunk. This procedure continues until the current FP chunk is full (i.e. capacity c of FP chunk is reached). The FP chunk is then fed to the progressive update component to oversample the FP instances in order to generate artificial samples. The oversampling component is later described in this Section. The set of FP instances together with the set of artificial instances are then utilised to update the pre-generated models. Let N represent the pre-generated set of genuine normal instances, and let A_s represent the set of artificial instances generated for FPchunk$_s$. Hence, the pre-generated models are retrained on R=N∪ FPchunk$_s$ ∪ A_s. Similarly, this process is repeated progressively for each accumulated FP chunk.

The rationale behind the progressive update method is not simply to retrain the models with new instances, but also to enrich the models with recently detected FPs and synthetically generated artificial samples *close (not replicates)* to FPs. Therefore, the decision boundary in each model adapts to the falsely detected behaviour and reduces the chances of FPs in the upcoming testing instances.

The idea of continuous monitoring to investigate flagged alarms to identify FPs has been applied in [61]. The authors defined cumulative recall measure based on a *daily budget*. The term *daily budget* refers to the maximum number of alarms an analyst can investigate per day to judge whether they are TPs or FPs.

Oversampling Component The method of updating pre-generated models with FP instances in the FP chunks may not have a significant influence on the decision boundary. However, the oversampling of FP instances generates artificial instances, and enriches the updated model with recently acquired FP instances, as well as normal artificial samples. In this way, the recently acquired normal behaviour of a user or a community will be well represented, and in turn will trigger the base method to adapt the model's decision boundary.

The oversampling component is in charge of generating artificial samples from the FP instances upon the acquirement of each FPchunk$_s$. The task of allocating the number of samples to be generated for each FP instance in the FPchunk$_s$ depends on the degree of outlierness of each FP instance. Thus, the number of samples to be generated varies among FP instances. We utilise a density-based method, namely, Local Outlier Factor (LOF) [113], to calculate the local outlier factor (score) lof_N^t for each FP instance FP^t in acquired FPchunk$_s$ with respect to the k_{LOF} nearest neighbours from only the set of genuine normal class instances N. lof_N^t is tuned for k_{LOF}=$\sqrt{1 + card(N)}$ (thumb-rule), where the radicand $1 + card(N)$ represents the number of instances utilised to calculate lof_N^t. The motivation to integrate LOF to calculate the anomaly score for FP instances, despite iForest can provide it, is that LOF can be used independently of the base method (ocsvm or iForest).

Let $perclof_N^t$ represent the percentile rank for each FP^t compared to the set of normal instances N. In Fig. 6.3, we represent the genuine normal instances N by blue filled circles. We define two *types of FP instances*, where each type is oversampled based on its degree of outlierness lof_N^t:

- *Outlier instance*: represented by a solid-line red empty circle. Each FP instance FP^t is considered an outlier instance, if it has a **high** lof_N^t value; located far away from the genuine normal instances N. For example, if $perclof_N^t$=90, this means that the lof_N^t for FP^t is greater than 90% of the normal instances N.
- *Safe instance*: represented by a solid-line blue empty circle. Each FP instance FP^t is considered a safe instance, if it has a **low** lof_N^t value, located at or near the borderline of genuine normal instances N.

In Fig. 6.3, the red dashed circles represent the artificial samples generated for FP outlier instances, while the blue dashed circles represent the artificial samples for FP safe instances. The idea is that safe instances are given more chance to generate artificial samples around them, while the outlier instances are given less chance. This gives the update component more conservative control on the adaptation of the decision boundary. Oversampling more safe instances than outlier instances safeguards the system from fast movement of the decision boundary due to outliers. Otherwise, more False Negatives (FN) (i.e. anomalous instances predicted as normal) will be in the upcoming FP chunks.

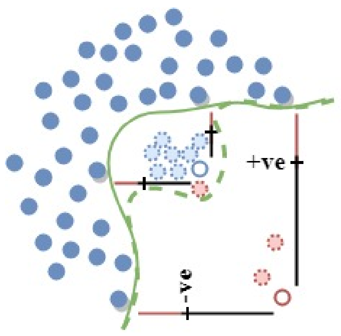


FIGURE 6.3: Artificial oversampling of FP instances over two features. The blue filled circles represent the genuine normal instances N, the solid-line red empty circle represents an FP outlier instance, and the solid-line blue empty circle represent an FP safe instance. The red dashed circles represent the artificial samples generated for FP outlier instances, and the blue dashed circles represent the artificial samples for FP safe instances. The solid-line green boundary represents the decision boundary of the pre-generated model, and green dashed decision boundary represents the adapted decision boundary of the updated model.

Let $perc.over$ represent the percentage of artificial samples to be generated, and let $numS$=$(perc.over/100) \times c$ represent the number of artificial samples to be generated. The process of generating artificial samples associated to each FP^t $\in FPchunk_s$ instance is executed feature-wise over a number of iterations.

Recall that $x_f^{t'}$ represents the value of the f^{th} feature of X$^{t'}$ $\in N$ at a session slot t'. Likewise, let p_f^t represent the value of the f^{th} feature of FP^t at a session slot t, given that FP^t=$\{p_1^t, p_2^t, ..., p_m^t\}$. For each feature $f; 1 \preceq f \preceq m$, we find the nearest neighbour $x_f^{t'}$ of $X^{t'} \in N$ for p_f^t of FP^t. In other words, we search the set of normal instances N at the level of feature f only, and we find the closest feature $x_f^{t'}$ for p_f^t. At the level of feature f, there exists two directions: positive ($+ve$), and negative ($-ve$). Thus, p_f^t may have (1) only $+ve$ neighbours from the set N, (2) only $-ve$ neighbours, or (3) both $+ve$ neighbours and $-ve$ neighbours. We define the positive ($+ve$) nearest neighbour and the negative ($-ve$) nearest neighbour as follows:

Definition 6.3.2: ***Positive nearest neighbour*** A $+ve$ nearest neighbour is the closest $x_f^{t'}$ of $X^{t'} \in N$ for p_f^t of FP^t, such that $x_f^{t'}$ is located in the $+ve$ direction to p_f^t.

Definition 6.3.3: ***Negative nearest neighbour*** A $-ve$ nearest neighbour is the closest $x_f^{t'}$ of $X^{t'} \in N$ for p_f^t of FP^t, such that $x_f^{t'}$ is located in the $-ve$ direction to p_f^t.

Fig. 6.4 illustrates generating artificial samples of FP instances over one feature (i.e. one dimension). Let the blue filled circle represent $x_f^{t'}$, the blue empty circle represent p_f^t, and the blue dashed circle represent an artificial feature value a_f^t associated to p_f^t. The process of generating an artificial feature value a_f^t is executed as follows:

- If p_f^t has only a $+ve$ nearest neighbour $x_f^{t'}$ at the level of feature f (Fig. 6.4.1), then a_f^t is calculated in the $+ve$ direction along the segment joining p_f^t and $x_f^{t'}$ according to Eq. 6.1, such that $dir{=}+1$.
- If p_f^t has only a $-ve$ nearest neighbour $x_f^{t'}$ at the level of feature f (Fig. 6.4.2), then a_f^t is calculated in the $-ve$ direction along the segment joining p_f^t and $x_f^{t'}$ according to Eq. 6.1, such that $dir{=}-1$.
- If p_f^t has both a $+ve$ nearest neighbour and a $-ve$ nearest neighbour at the level of feature f (Fig. 6.4.3), then a_f^t can be calculated in the $+ve$ or $-ve$ direction. A random direction dir is selected at each iteration, and a_f^t is calculated in the selected direction dir according to Eq. 6.1.

$$a_f^t{=}p_f^t \; + dir \times rand(0 : \lambda \times dist(p_f^t, x_f^{t'})) \tag{6.1}$$

where $dist(p_f^t, x_f^{t'})$ represents the distance between an FP feature value p_f^t and the nearest neighbour $x_f^{t'}$.

Note that λ, tuned for λ=0.8, denotes a parameter that controls the distance permitted to generate artificial features along the segment joining p_f^t and $x_f^{t'}$. The value of λ is in the range $]0.5, 1[$. The rationale behind this is to generate the artificial samples a bit closer to the FP instances and not the normal instances, so that the adapted decision boundary is influenced by these samples.

Consequently, an artificial sample $A^t{=}\{a_1^t, a_2^t, ..., a_f^t\}$ associated with an FP^t instance is generated at each iteration. The steps described are repeated for a number of iterations until $numS$ of artificial samples are generated.

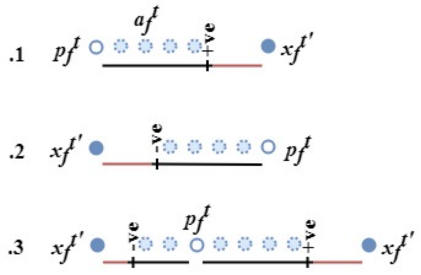


FIGURE 6.4: Artificial oversampling of FP instances over one feature. The blue filled circle represents a genuine normal feature value $x_f^{t'}$, the blue empty circle represents an FP feature value p_f^t, and the blue dashed circle represents an artificial feature value a_f^t associated to p_f^t. Fig. 6.4.1 illustrates the case where p_f^t has only a $+ve$ nearest neighbour $x_f^{t'}$. Fig. 6.4.2 illustrates the case where p_f^t has only a $-ve$ nearest neighbour $x_f^{t'}$. Fig. 6.4.3 illustrates the case where p_f^t has two nearest neighbours in both directions.

6.4 Experiments

The experiments are conducted on the CMU-CERT data sets on Windows Server 2016 on Microsoft Azure (RAM $140GB$, OS $64 - bits$, CPU Intel Xeon $E5 - 2673v3$). We implemented the experiments in R environment ($R - 3.4.1$) using the following packages: **e1071** package for ocsvm, **isofor** package for iForest, **caret** package [114] for evaluation, **Rlof** package for LOF in the oversampling method, and **MASS** package for the Wilcoxon Ranked test.

6.4.1 Experimental Setup

In Table 6.1, we define a variety of experiments performed using the aforementioned anomaly detection methods. Each experiment is set up to ocsvm or iForest base methods with or without the following: ensemble method, progressive update method, and oversampling method. To evaluate the methods, we performed 10 fold cross-validation on each of the utilised communities.

The experiments are tuned for different values of parameters. *Note that an extensive number of experiments was done to select the presented tuning values for the parameters. The values were selected based on the experiments achieving the best performance in terms of the evaluation measures described below.*

ocsvm is evaluated for the $kernel$ values: Linear L, Polynomial P, and Radial R. On the other hand, iForest is presented for different values of anomaly score

TABLE 6.1: Definition of Experiments.

E-ocsvm	E-iForest	**E**nsemble
ocsvm-U	iForest-U	progressive **U**pdate
E-ocsvm-U	E-iForest-U	**E**nsemble+**U**pdate
ocsvm-OU	iForest-OU	**U**pdate+**O**versampling
E-ocsvm-OU	E-iForest-OU	**E**nsemble+**U**pdate+**O**versampling

threshold τ=$\{0.35, 0.4, 0.45\}$, given the number of iTrees nt=20 and the subsample size ψ. Note that no anomalous instances were isolated for $\tau > 0.5$ over the utilised CMU-CERT data sets. This reflects the complex patterns of malicious insider threat behaviour in the data set, and the high resemblance of anomalous behaviour with normal behaviour. Although the authors in [50] pointed out the efficiency of subsampling, ψ is tuned for the whole sample size without extracting a subsample of the data set. It is either set up to $card(N)$ over the whole normal training data set, or to $card(C_i)$ over each cluster in the ensemble case. The rationale of using the whole sample size in both cases is related to the progressive update method. It assures that all the FP instances in the progressively acquired FP chunks are used to update the iForest model.

Regarding the ensemble method, we tuned the number of clusters k over a set of arbitrarily chosen small values k=$\{2, 4\}$, as the goal is to detect the anomalous trapper instances. However, the results for only k=2 are reported, due to revealing better performance in terms of detecting most of the malicious insider threats in the data sets. As mentioned in Section 5.4.2, the literature demonstrated the effectiveness of k-means clustering for small values of k when applied for class decomposition [104], [109]. The capacity of FP chunks is presented over the values c=$\{20, 40, 60, 80, 100\}$ to evaluate the effectiveness of updating the model with *early-acquired-early-updated FPs*. For the oversampling method, the oversampling percentage is tuned for only $perc.over$=200 to test the influence of applying oversampling to the FP chunks on the progressive update method.

6.4.2 Evaluation Measures

The ultimate aim of the proposed framework is to detect all the malicious insider threats (not necessarily all behaviours), while reducing the number of FPs (all false alarms). We utilised the previously defined measures in Chapter 5 to evaluate the experiments. First, we recall the *default* versions of measures that are evaluated per instance (behaviour): **FP** designates the number of normal instances (behaviours) that are detected as anomalous instances. FP is evaluated with respect to the whole testing set (not per chunk); and TN designates the number of normal instances predicted as normal.

Second, we recall the *refined* versions of measures that are evaluated per threat: **TP**$_T$ is used instead of TP to evaluate the number of threats detected by the system among all the P$_T$ malicious insider threats. TP$_T$ is incremented if at least one anomalous instance (behaviour associated to the threat) is predicted as anomalous (the aim of any-behaviour-all-threats –*threat hunting*); and FN$_T$ is used instead of FN to evaluate the number of insider threats not detected.

As a result, **F1** measure is defined based on the values of the above defined measures. The maximum of traditional F1 measure of 1 is not expected here, given the refined definition of the measures. Dependent on the data distribution, F1 measure closer to 0.5 reveals *good* performance. More detailed information about the evaluation measures can be found in Section 5.4.3.

6.4.3 Results and Discussion

In this Section, we present the results of ocsvm experiments and iForest experiments in terms of the pre-defined evaluation measures. We then show the merit of the proposed framework according to the following targets:

- Testing the statistical significance of the results using the Wilcoxon Signed-Rank Test;
- Comparing ocsvm experiments and iForest experiments;
- Assessing the influence of the proposed components on the framework; and
- Assessing the time efficiency of the progressive update component.

TABLE 6.2: Minimum FP over communities associated with (1) the $kernel$ and FP chunk capacity c which achieved the minimum FP for ocsvm based experiments, and (2) the anomaly score threshold τ and c which achieved the minimum FP for iForest based experiments. Note that the parameter combination(s) which achieved the minimum FP are listed. The results are displayed as $\langle$FP $kernel, c\rangle$ for ocsvm based experiments and $\langle$FP $\tau, c\rangle$ for iForest based experiments.

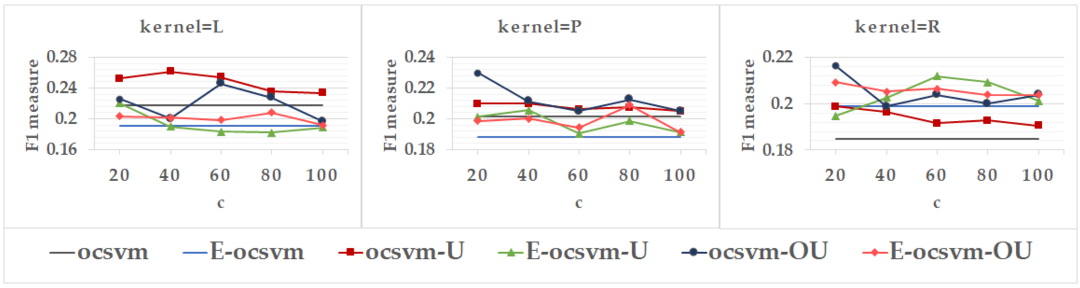
community	ocsvm	E-ocsvm	ocsvm-U	E-ocsvm-U	ocsvm-OU	E-ocsvm-OU
com-P	106 L	97 L	**76** L,40	88 L,20	83 L,60	95 L,80
com-S	120 R	90 P	99 L,20	**89** P,60	109 P,20	**90** P,20
com-I	149 R	100,$\{L, P\}$	146 R,$\{20, 40\}$	**52** P,20	132 P,20	97 P,20
community	**iForest**	**E-iForest**	**iForest-U**	**E-iForest-U**	**iForest-OU**	**E-iForest-OU**
com-P	88 0.45	66 0.45	85 0.45,$\forall c \setminus 100$	**50** 0.45,20	84 0.45,40	**57** 0.45,20
com-S	85 0.45	73 0.45	82 0.45,$\{20, 100\}$	**49** 0.45,20	78 0.45,100	70 0.45,20
com-I	80 0.45	69 0.45	68 0.45,60	**43** 0.45,20	72 0.45,60	53 0.45,20

TABLE 6.3: Maximum TP_T of detected insider threats over communities associated with (1) the $kernel$ and FP chunk capacity c which achieved the maximum TP_T for ocsvm based experiments, and (2) the anomaly score threshold τ and c which achieved the maximum TP_T for iForest based experiments. Note that the parameter combination(s) which achieved the maximum TP_T are listed. The results are displayed as $\langle$FP $kernel, c\rangle$ for ocsvm based experiments and $\langle$FP $\tau, c\rangle$ for iForest based experiments.

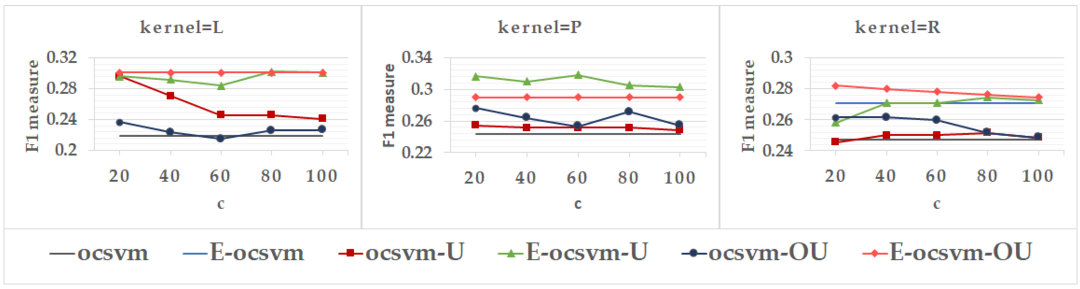
community	ocsvm	E-ocsvm	ocsvm-U	E-ocsvm-U	ocsvm-OU	E-ocsvm-OU
com-P	**17** P	16 R	**17** P,$\forall c$	16 R,$\forall c \setminus 20$	**17** P,$\forall c$	16 R,$\forall c$
com-S	**22** P	21 R	**22** P,$\forall c$	**22** P,$\{20 - 40\}$	**22** L,$\{20, 60\}$ P, 80	21 R, $\forall$
com-I	**12** $\forall kernel$	**12** R	**12** $\forall kernel$,$\forall c$	**12** R,$\forall c$	**12** $\forall kernel$,$\forall c$	**12** P, 20 R,$\forall c$
community	**iForest**	**E-iForest**	**iForest-U**	**E-iForest-U**	**iForest-OU**	**E-iForest-OU**
com-P	16 0.35	16 0.35	16 0.35,$\forall c$	16 0.35,$\forall c$	**17** 0.35,$\{20 - 60\}$	**17** 0.35,20
com-S	21 0.35	21 0.35	21 0.35,$\forall c$	21 0.35,$\forall c$	**22** 0.35,$\forall c \setminus 60$	**22** 0.35,$\forall$
com-I	**12** $\forall \tau$	**12** $\forall \tau$	**12** $\forall \tau$,$\forall c$	**12** $\forall \tau$,$\forall c$	12 $\forall \tau$,$\forall c$	**12** $\forall \tau$,$\forall c$

6.4.4 Results

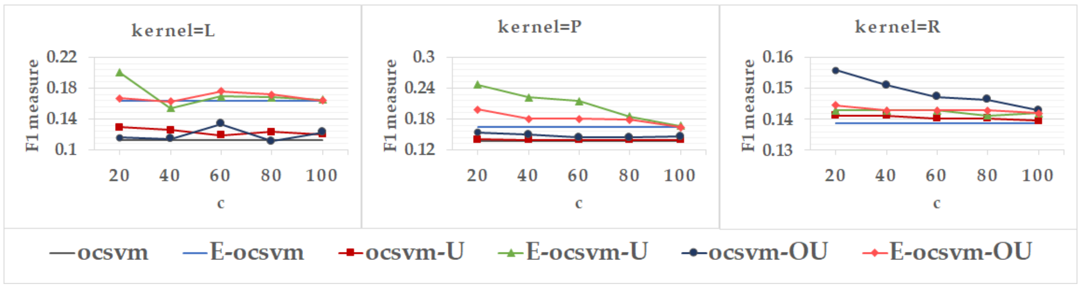
Tables 6.2 and 6.3 present the minimum FP and the maximum TP_T attained respectively for ocsvm and iForest based experiments over the three communities. The tables report $kernel$ and FP chunk capacity c associated with the minimum FP or maximum TP_T attained for ocsvm based experiments. On the other hand, the tables present anomaly score threshold τ and c associated with the minimum FP or maximum TP_T attained for iForest based experiments. This allows the identification of the superior $kernel$ or τ, and analysing the influence of FP chunk capacity c on the progressive update method.



(A) com-P.



(B) com-S.



(C) com-I.

FIGURE 6.5: The variation of F1 measure as a function of FP chunk capacity c for ocsvm based experiments over communities. The x-axis represents the FP chunk capacity c=$\{20, 40, 60, 80, 100\}$, and the y-axis represent the values of F1 measure, and the graphs in Fig. 6.5a, Fig. 6.5b, and Fig. 6.5c represent the results for $kernel$={ L, P, R }.

6.4.4.1 ocsvm Experiments

Fig. 6.5 presents the variation of F1 measure as a function of FP chunk capacity c for ocsvm based experiments over the communities. The results are reported with respect to $kernel$ values.

Over com-P, ocsvm-U outperforms ocsvm, as well as all other ocsvm based experiments in terms of F1 measure, FP, and TP$_T$. ocsvm-U achieves the maximum F1=**0.261**; $kernel$=L while reducing the false positives to the minimum FP=**76**; $kernel$=L compared to FP=106 in ocsvm. Knowing that P$_T$=17, ocsvm-U detects all the malicious insider threats TP$_T$=17; $kernel$=P without missing any threat.

Over com-S, E-ocsvm-U and E-ocsvm-OU report better performance for all $kernels$, where E-ocsvm-U achieves the maximum F1=**0.318**; $kernel$=P with the minimum FP=**89**; $kernel$=P compared to FP=120 in ocsvm. E-ocsvm-OU follows it with F1=0.300; $kernel$=L and FP=90;P. However, E-ocsvm-U attains the maximum TP$_T$=22; $kernel$=P out of P$_T$=22, unlike the latter that missed one threat.

Over com-I, E-ocsvm-U reports the best performance in terms of all measures compared to other ocsvm based experiments. It achieves the maximum F1=**0.246**; $kernel$=P, while reducing the number of false positives to FP=**52**; $kernel$=P knowing that FP=149 in ocsvm. Furthermore, it attains TP$_T$=12 out of P$_T$=12 (same TP$_T$ as for other experiments).

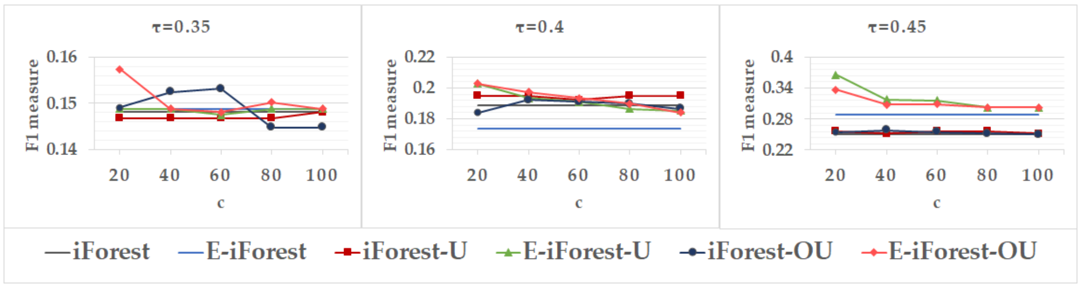
6.4.4.2 iForest Experiments

Fig. 6.6 presents the variation of F1 measure as a function of FP chunk capacity c for iForest based experiments over the communities. The results are reported with respect to anomaly score threshold τ values.

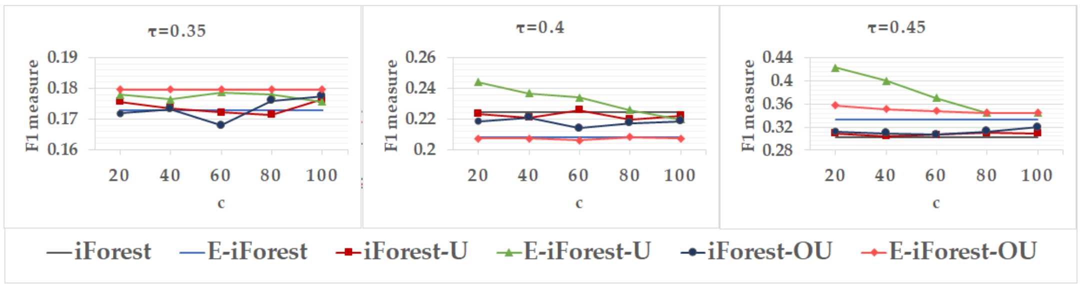
Over com-P, E-iForest-U and E-iForest-OU gave a significant boost to F1 measure for τ=0.45, where it reaches F1=**0.365**,0.337 respectively. E-iForest-U reduces the number of false positives to the minimum FP=**50**; τ=0.45 compared to FP=**88** in iForest, however, it detects TP$_T$=16; τ=0.35, thus missing one threat. On the other hand, E-iForest-OU reaches a minimum FP=57; τ=0.45, while detecting all the malicious insider threats
TP$_T$=17; τ=0.35.

Over com-S, E-iForest-U reports the best performance in terms of F1 and FP compared to other iForest based experiments. It achieves the highest F1=**0.422**; τ=0.45, and the minimum FP=**49**; τ=0.45 compared to FP=**85** in iForest. However, it reports TP$_T$=21; τ=0.35, thus missing the detection of one malicious insider threat. It is worth to note that the iForest based experiments with oversampling method (with or without the ensemble method) attain to detect all the threats for τ=0.35

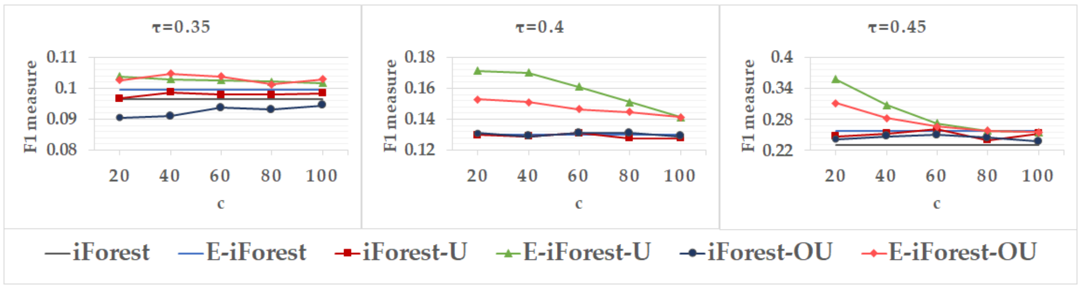
Over com-I, E-iForest-U shows the best performance in terms of all measures. It achieves the maximum F1=**0.358**; τ=0.45, while reducing the number of false positives to FP=**43**; τ=0.45 compared to FP=**80** in iForest. It also attains the maximum TP$_T$=12; $\forall\tau$ (same TP$_T$ for other experiments).



(A) com-P.



(B) com-S.



(C) com-I.

FIGURE 6.6: The variation of $F1$ measure as a function of FP chunk capacity c for iForest based experiments over communities. The x-axis represents the FP chunk capacity c=$\{20, 40, 60, 80, 100\}$, and the y-axis represent the values of F1 measure, and the graphs in Fig. 6.6a, Fig. 6.6b, and Fig. 6.6c represent the results for τ=$\{00.35, 0.4, 0.45\}$.

6.4.5 Statistical Significance: Wilcoxon Signed-Rank Test

To test the significance of the results, we use the Wilcoxon Signed-Rank Test that compares each pair of experiments. Each ocsvm-based experiment and iForest-based experiment is compared with base ocsvm and base iForest respectively. Table 6.4 tabulates the *p-value* calculated for each experiment at .05 significance level. For instance, E-ocsvm (first cell) shows the p-value for the ⟨E-ocsvm, ocsvm⟩ pair; a result of 'the values of F1 measure for E-ocsvm $\forall kernel$,$\forall c$' compared with 'the values of F1 measure for ocsvm $\forall kernel$,$\forall c$.'

With respect to ocsvm, all ocsvm-based experiments, except E-ocsvm ($0.1073 >$ .05), are significantly different from base ocsvm where *p-value* $< .05$. With respect to

TABLE 6.4: Wilcoxon Signed-Rank Test at .05 significance level. The p-value is calculated for each ocsvm based experiment or iForest based expeiment compared with base ocsvm or base iForest respectively. For instance, E-ocsvm (first cell) shows the p-value for the ⟨E-ocsvm, ocsvm⟩ pair; a result of 'the values of F1 measure for E-ocsvm $\forall kernel$,$\forall c$' compared with 'the values of F1 measure for ocsvm $\forall kernel$,$\forall c$.'

E-ocsvm	ocsvm-U	E-ocsvm-U	ocsvm-OU	E-ocsvm-OU
1.073e-01	6.101e-09	8.243e-06	8.587e-07	1.687e-06
E-iForest	iForest-U	E-iForest-U	iForest-OU	E-iForest-OU
2.719e-01	1.945e-04	4.104e-08	**0.1165**	1.986e-05

base iForest, all iForest-based experiments are significantly different from base iForest, except for E-iForest and iForest-OU. The results were predictable, because both E-ocsvm and E-iForest show low performance in terms of the evaluated measures.

6.4.6 Comparing ocsvm Experiments and iForest Experiments

Finally, it is noteworthy that base iForest flagged a lower number of false alarms and reported a higher F1 measure compared to base ocsvm. Moreover, the best performing iForest-based experiments over each community achieved the minimal FP compared to ocsvm-based experiments. For instance, E-iForest attains the minimum FP=50; τ=0.45, c=20, while ocsvm-U attains FP=76; $kernel$=L, c=40, compared to FP=106 in base ocsvm over com-P. Over com-S, E-iForest-U reaches the minimum FP=49; τ=0.45, c=20, while E-ocsvm-U and E-ocsvm-OU attain FP=89; $kernel$=P, c=60 and FP=90 respectively. Finally, E-iForest-U achieves the minimum FP=43; τ=0.45, c=20, compared to FP=52; $kernel$=P, c=20 in E-ocsvm-U over com-I. In these experiments, ocsvm works better for $kernel$=$\{L, P\}$, and iForest shows its best performance for τ=0.45. Moreover, the less the capacity (size) c of FP chunk, the better the performance of the progressive update method. This emphasises the effectiveness of updating the model with *early-acquired-early-updated FPs* in reducing the number of FPs.

The ensemble-ocsvm approach proposed by Parveen et al. [22] reports a percentage of FP rate (FP=30%) on the 1998 DARPA Intrusion Detection data set [71]. However, our framework reports approximately a half of this value, where %FP varies between (14%-18%) over all utilised communities in the $r5.2$ CMU-CERT data sets.

It is worth noting that the provided comparison is only indicative of the performance given that the ensemble-ocsvm and our framework were evaluated on different data sets.

6.4.7 Influence of Proposed Components on the Framework

Based on the above extensive experiments carried out to evaluate the influence of the different components on the proposed framework, we sum up what follows. ocsvm-U, E-ocsvm-U, and E-ocsvm-OU outperform other ocsvm-based experiments in terms of F1 measure, FP, and TP_T for all communities. On the other hand, E-iForest-U and E-iForest-OU outperform other iForest-based experiments for all communities.

The importance of the clustering component and the progressive update component as a whole is quite evident, where the number of FPs reached its minimal in experiments that utilised the ensemble method and the progressive update method with FP chunks (e.g. E-ocsvm-U, E-ocsvm-OU, E-iForest-U, E-iForest-OU). We deduce that the effectiveness of the proposed framework relies on the joint collaboration of both components. The use of the clustering component solely, in E-ocsvm and E-iForest, did not improve the performance significantly in terms of the evaluated measures. The Wilcoxon test supported this as revealed in Table 6.4. However, the joint use of progressive update method with ensemble method showed better performance in terms of reducing the number of FPs significantly and achieving higher F1 measure.

Furthermore, we can infer from Table 6.3 the support for the oversampling method that detects all the malicious insider threats over all communities without missing any threat. Also, it is quite remarkable that E-ocsvm-OU and E-iForest-OU show comparable performance to E-ocsvm-U and E-iForest over com-S and com-P respectively in terms of F1 measure and FP. This reveals the potential of the oversampling component.

6.4.8 Time Efficiency of Progressive Update Component

Regarding the time complexity, the merit of the proposed progressive update method is its time efficiency. Let ta_s represent the time to accumulate an FPchunk$_s$. We argue that the time to update the ensemble models with the current FPchunk$_s$ is much less than the time to accumulate FPchunk$_{s+1}$ (ta_{s+1}). This ensures that the progressive update of the models is synchronised with the accumulation of sequentially acquired FP chunks.

Recall that, an instance X^t from the acquired test instances is accumulated in the FP chunk, if X^t is declared FP. The FP chunk is progressively accumulated until the capacity c is reached. Hence, the time to accumulate an FP chunk FPchunk$_{s+1}$ actually depends on, (1) n: number of acquired test instances after FPchunk$_s$ until capacity c of FPchunk$_{s+1}$ is reached, and (2) $slot$: period of a session slot for a test instance. Analytically, ta_{s+1}=$n \times slot$.

Consider c=20 and $slot$=4 hours. If n=20, this means that c=20 FPs are accumulated in FPchunk$_{s+1}$ after 20 test instances are acquired. Thus, ALL the n acquired test instances are actually FPs. In this case, ta_{s+1}=20 $\times$ 4=80 hours (approximately 3 days). This case represents the *worst case scenario*, where the next FPchunk$_{s+1}$ is accumulated while 80 hours (3 days) are available to update the models with the current FPchunk$_s$. 80 hours is more than enough to update the models using any state-of-the-art cloud facility. In fact, the time needed to update the ensemble models with the current FPchunk$_s$ is actually much less than 80 hours, which confirms the time efficiency of the method.

In the aforementioned worst case scenario, the progressive update is required to run after 80 hours. However, in a typical scenario where an FP chunk is accumulated after $n > c$ number of test instances is acquired, the progressive update will run much less frequently.

6.5 Summary

In this Chapter, we address the shortcoming of the high number of FPs based on the availability of only data for normal behaviour, with no previously logged insider incidents.

We propose an adaptive one-class ensemble-based anomaly detection framework with a progressive artificial oversampling method of FPs in class decomposed data. First, this framework introduces a class decomposition method to cluster the normal class data in order to detect anomalous *trapper* instances. The result is an ensemble of k base methods (ocsvm or iForest) trained over the k clusters. Unlike the proposed framework, CD-AMOTRE in Chapter 5 applied class decomposition to weaken the effect of the normal class to handle the class imbalance data problem (transform two-class into multi-class classification). Furthermore, unlike targeting trapper instances as anomalous instances in the proposed framework, the -TRE part in Chapter 5 removed the trapper instances which would trap the classifier.

Second, a progressive update method updates the pre-generated models with progressively acquired FP chunks. This allows the models to benefit from already declared FPs and to progressively adapt to the change in user's behaviour in order to reduce FPs in upcoming data.

Third, an oversampling procedure is integrated to the progressive update method. The merit of oversampling is that it is outlier-aware, which allows to intelligently update and enrich the models with artificial samples of FPs, thus avoiding data overfitting. Unlike the proposed framework, the oversampling method in Chapter 5 is applied on anomalous instances, not normal instances (FPs).

We evaluate the proposed framework in a variety of experiments (with and without the following methods: class decomposition/progressive update/oversampling) using ocsvm and iForest. The results demonstrated the effectiveness of the joint collaboration of the methods on the performance in terms of higher F1 measure, lower FP (the best minimum was 14.4%), and detecting *all* malicious insider threats. iForest-based experiments report a better F1 and a lower FP compared to that of ocsvm.

Chapter 7

Streaming Anomaly Detection for Insider Threat Detection

7.1 Introduction

A data stream is a continuous acquisition of data generated from various source(s) in a dynamic environment, typically in a high velocity, leading to accumulation of large volumes of data. This characterisation leads to a typical Big Data computational problem. The dynamic nature of a data stream imposes a change in the data over time. In real-time data streaming, a change refers to an anomalous data that deviates from the normal baseline (e.g. credit card fraud, network intrusion, cancer, etc.). The ultimate aim of such stream mining problems is to detect anomalous data in real-time.

The absence of prior knowledge (no historical database i.e. low data maturity) is often a complication of a real-time stream mining problem. The anomaly detection system is required to employ unsupervised learning to construct an adaptive model that continuously (1) updates itself with new acquired data, and (2) detects anomalous data in real-time. The system usually acquires data as segments and identifies the *outliers* in the segment as anomalous. An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism [119]. To detect outliers in a stream mining problem, several approaches have been proposed, nevertheless, unsupervised clustering has been successfully applied to identify the patterns in the data and spot outliers

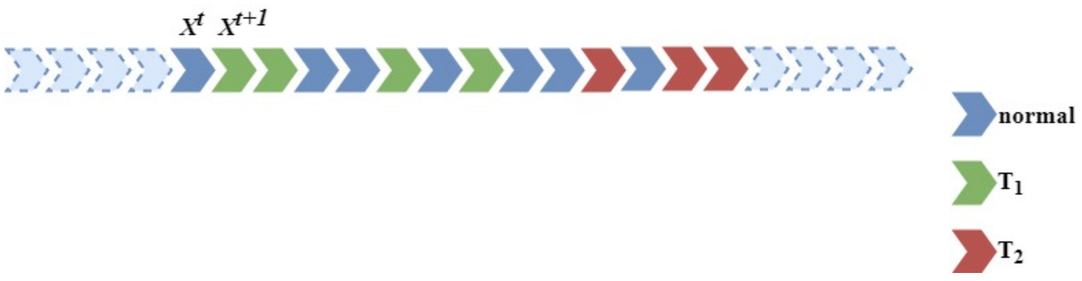


FIGURE 7.1: Continuous data stream of behaviours. Each arrow denotes a behaviour (instance) X^t executed by a user at a specific period of time t. The blue arrows represent the normal behaviours, and the green arrows and the red arrows represent the anomalous behaviours that belong to the malicious insider threats T_1 and T_2 respectively. To detect a malicious insider threat T_1, it is required to detect **any** green behaviour X^t.

[120]. In this Chapter, we tackle the insider threat problem to detect malicious insider threats (outliers) in real-time. With the absence of labelled data (no previously logged activities executed by users in an organisation), the insider threat problem poses a challenging stream mining problem.

As described in Section 3.3, we formulate this work with the aim to detect **any-behaviour-all-threat**; it is sufficient to detect **any** anomalous behaviour in **all** malicious insider threats. The design of the proposed approach with such a relaxing condition contributes in reducing the frequent false alarms.

Fig. 7.1 illustrates a continuous data stream of behaviours (instances) including normal behaviours and anomalous behaviours. Each arrow denotes a behaviour (instance) X^t executed by a user at a specific period of time t. Let the blue arrows represent the normal behaviours. Let the green arrows and the red arrows represent the anomalous behaviours that belong to the malicious insider threats T_1 and T_2 respectively. To detect a malicious insider threat T_1, it is required to detect **any** green behaviour X^t. Hence, it is essential to detect **any** of the anomalous behaviours per malicious insider threat; the aim to detect any-behaviour-all-threat. Note that the proposed approach may detect **more than one** behaviour that belong to a malicious insider threat, nevertheless, the ultimate aim is to detect one anomalous behaviour per threat.

The 'behaviour-based' streaming anomaly detection approaches reviewed in Section 2.3.2.3 have shown merit in addressing the insider threat detection problem, however, they do suffer from high false alarms. For example, ensemble-GBAD [63] reports a significant high FP rate=54%, ensemble-ocSVM reports FP rate =31% in [63], and ensemble-ocsvm further reports FP rate=**24%** in [22] (refer to Table 2.2).

To address the shortcoming of the high number of false alarms, we propose a streaming anomaly detection approach, namely Ensemble of Random subspace Anomaly detectors In Data Streams (E-RAIDS). We presented a preliminary version of E-RAIDS (termed Random Subspace Outlier detector – RandSubOut) in [25]. E-RAIDS is built on top of established outlier detection techniques (Micro-cluster-based Continuous Outlier Detection –*MCOD*– or Anytime Outlier detection –*AnyOut*–) that employ clustering over continuous data streams. The merit of E-RAIDS is its capability to detect malicious insider threats in real time (*based on the definition of real-time in terms of window iterations as discussed in Section* 7.4).

E-RAIDS learns an ensemble of p outlier detection techniques (either MCOD or AnyOut), such that each model of the p models learns on a random feature subspace. The acquired data is accumulated in a temporary *buffer* of predefined capacity (equals to a fixed window size). At each window iteration, each of the p models in the ensemble is updated with the acquired data, and local outliers are identified in the corresponding feature subspace.

Let p=$m + 1$ denote the number of feature subspaces selected randomly, such that m is set to the number of features (dimensions) of the feature space F in the community data set. m represents the number of feature pairs (i.e. 2 features per subspace), and 1 represents the whole feature space (i.e. all features). In this way, E-RAIDS is capable to detect *local outliers* in the m feature pairs, as well as *global outliers* in the whole feature space. Hence, E-RAIDS employs the idea of random feature subspaces to detect local outlier(s) (anomalous behaviour(s)), which might not be detected over the whole feature space. These anomalous behaviour(s) might refer to malicious insider threat(s).

E-RAIDS introduces two factors: (1) a survival factor vf_s, which confirms whether a **s**ubspace votes for an alarm, if outlier(s) survive for a vf_s number of window iterations; and (2) a vote factor vf_e, which confirms whether an alarm should be flagged, if a vf_e number of subspaces in the **e**nsemble vote for an alarm. E-RAIDS employs an aggregate component that aggregates the results from the p feature subspaces, in order to decide whether to generate an alarm. The rationale behind this is to reduce the number of false alarms.

The main contributions of this chapter are summarised as follows:

- an ensemble approach on random feature subspaces to detect local outliers (malicious insider threats), which would not be detected over the whole feature space;
- a survival factor that confirms whether outlier(s) on a feature subspace survive for a number of window iterations, to control the vote of a feature subspace;
- an aggregate component with a vote factor to confirm whether to generate an alarm, to address the shortcoming of high number of FPs;
- a thorough performance evaluation of E-RAIDS-MCOD and E-RAIDS-AnyOut, validating the effectiveness of voting feature subspaces, and the capability to detect (more than one)-behaviour-all-threat (Target 2) in real-time (Target 1).

The rest of this Chapter is organised as follows. Section 7.2 (1) reviews the techniques that utilised clustering to detect outliers, and (2) gives background on established continuous distance-based outlier detection techniques. Section 7.3 presents the proposed streaming anomaly detection approach, namely E-RAIDS, for insider threat detection. Experiments and results are discussed in Section 7.4. Finally, Section 7.5 summarises the Chapter.

7.2 Background

The absence of labelled data (i.e. low data maturity) means that an organisation has no previously logged activities executed by users (no historical database). We address the absence of prior knowledge using unsupervised streaming anomaly detection built on top of established outlier detection techniques.

Clustering has been successfully applied to identify patterns in data for outlier detection [120]. Continuous acquisition of data generated from various sources defines the streaming environment of the insider threat problem. Several clustering methods have been proposed to handle the streaming environment. The state-of-the-art presents two primitive clustering methods: Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [121], and CluStream [122].

BIRCH is an incremental hierarchical clustering method that was first proposed for very large data sets. BIRCH incrementally and dynamically clusters acquired data instances. It maintains a tree of cluster features (information about clusters) that is updated in an iterative fashion [123]. Thereafter, BIRCH was applied to data stream clustering. BIRCH was the first to introduce the concepts of micro- and macro- clusters [124].

CluStream is a data stream clustering method that employed those two concepts for the clustering process: online micro-clustering, and offline macro-clustering. In the online phase, CluStream scans the acquired data instances and creates micro-clusters in a single pass to handle the big (unbounded) data stream. In the offline phase, CluStream only utilises the micro-clusters and re-clusters into macro-clusters [123].

From BIRCH to CluStream, the concept of data stream clustering is applied, despite the fact that BIRCH includes incremental processing of data instances. *Incremental clustering* processes one data instance at a time and maintains a predefined data structure (i.e. model) that is incrementally updated without the need for model reconstruction [125]. In fact, many incremental methods predate the data stream mining methods. The intrinsic nature of data streams requires methods which implement incremental processing of data instances, in order to handle the resource limitations (i.e. time and memory) [125]. But unlike earlier incremental clustering methods, *data stream clustering* methods require a more efficient time complexity to cope with high data rates [126]. Indeed, the literature stresses the importance of considering the inherent time element in data streams [125]. For instance, a typical data stream clustering method exhibits the temporal aspect of cluster tracking [125]. The dynamic behaviour of cluster over a data stream manifests in the evolving of existing clusters over time; the emergence of new clusters; and the removal of clusters based on a time stamp and/or size. Those cluster updates must be performed on the data structure (i.e. model) very efficiently [126]. Hence, the data stream clustering methods are not only incremental, but are also fast in terms of the inherent temporal aspects.

In this work, data stream clustering is used to underpin the outlier detection techniques for real-time anomaly detection. In the insider threat problem, it is vital to consider the temporal aspect, in order to detect malicious insider threats in real-time (based on the definition of real-time in terms of window iterations as discussed in Section 7.4). Hence, data stream clustering methods are more appropriate to be utilised in this work than typical incremental clustering methods.

In the following we review the techniques which utilised clustering to detect outliers. Thereafter, we shed light on two outlier detection techniques (MCOD and AnyOut) that employ *data stream clustering*. These two techniques are later utilised in the proposed framework.

We then give a background on distance-based outlier detection techniques, including a description and formalisation of the established techniques – MCOD and AnyOut.

7.2.1 Clustering for Outlier Detection

It is important to distinguish our objective – to optimise outlier detection – from that of a benchmark of clustering methods developed to optimise clustering (such as DBSCAN [127], BIRCH [121]). The outliers in these methods [121], [127] are referred to as noise, where they are usually tolerated or ignored. However, in our work, we refer to outliers as anomalous (suspicious) behaviour(s) that may be associated with malicious insider threats. Therefore, clustering is utilised to optimise the detection of outliers.

On the other hand, a benchmark of clustering methods to optimise outlier detection (such as LOF [113], CBLOF [128]) are developed. Breunig et al. [113] introduced the concept of Local Outlier Factor (LOF) to determine the degree of outlierness of an instance using density-based clustering. The locality of an instance (local density) is estimated by the distance to its k nearest neighbours. Thus, the LOF of an instance located in a *dense* cluster is close to 1, while higher LOF is assigned for other instances. It is worth noting that LOF was utilised in the approaches of Chapter 5, and Chapter 6. He et al. [128] present a measure, called Cluster-based Local Outlier Factor (CBLOF), to identify the physical significance of an outlier using a similarity function. The CBLOF of an instance is determined by (1) the distance to its nearest cluster (if it belongs to a small cluster), or (2) the distance to its cluster (if it belongs to a large cluster).

However, the aforementioned methods do not tackle outlier detection in *continuous* data streams. MCOD [126] and AnyOut [120] are two established outlier detection techniques that employ data stream clustering in *continuous* data streams. MCOD and AnyOut are utilised as building block outlier detection techniques for the proposed E-RAIDS approach. A detailed description and formalisation for these techniques can be found in Section 7.3.2.

7.2.2 Background on Distance-based Outlier Detection Techniques

The state-of-the-art presents one of the most widely employed techniques for anomaly detection, which are *distance-based* outlier detection techniques. According to the definition in [129], an instance X^t is an outlier, if there exists less than k number of neighbours located at a distance at most r form X^t.

In this work, we are interested in *continuous outlier detection* over a data stream, where recent instances arrive and previous instances expire. The demo paper [130] gives a comparison of four continuous distance-based outlier detection techniques: STream OutlieR Miner (STORM) [131]; Abstract-C [132]; Continuous Outlier Detection (COD) [126]; and Micro-cluster-based Continuous Outlier Detection (MCOD) [126]. COD and MCOD have $O(n)$ space requirements and a faster running time than the exact algorithms of both [131] and [132]. Note that since all algorithms are exact, they identify the same outliers [130]. According to [130], COD and MCOD demonstrate a more efficient performance compared to STORM and Abstract-C in terms of lower space and time requirements. Hence, the latter are excluded, but not COD and MCOD. Furthermore, based on the experimental evaluation in [126], MCOD outperforms COD over benchmark tested data sets. Hence, MCOD is selected to be utilised as a base learner in the proposed E-RAIDS approach.

The state-of-the-art presents a further continuous distance-based outlier detection technique, called Anytime Outlier detection (AnyOut) [120]. However, AnyOut has not been compared to the four techniques in the demo paper [130]. We select MCOD and AnyOut as base learners in E-RAIDS to compare their performance. It is worth noting that the previously mentioned distance-based outlier detection techniques are implemented by the authors of [130] in the open-source tool for Massive Online Analysis (MOA) [133].

In the following, a brief description of MCOD and AnyOut techniques is provided.

7.2.2.1 Micro-cluster-based Continuous Outlier Detection

Micro-cluster-based Continuous Outlier Detection (MCOD), an extension to Continuous Outlier Detection (COD), stems from the adoption of an event-based technique. The distinctive characteristic of MCOD is that it introduces the concept of evolving micro-clusters to mitigate the need to evaluate the range query for each acquired instance X^t with respect to all the preceding active instances. Instead, it evaluates the range queries with respect to the (fewer) centres of the micro-clusters. The micro-clusters are defined as the regions that contain inliers exclusively (with no overlapping). The micro-clustering is fully performed online [126].

Given that, the centre mc_i of a micro-cluster MC_i may or may not be an actual instance X^t; the radius of MC_i is set to $\frac{r}{2}$, such that r is the distance parameter for outlier detection; and the minimum capacity (size) of MC_i is $k+1$. Below we give a brief formalisation of MCOD. Let k represent the number of neighbours parameter. Let PD represent the set of instances that doesn't belong to any micro-cluster.

The micro-clusters (i.e. centres of micro-clusters) in MCOD are determined as described in [122]. In the initialisation step, seeds (with a random initial value) are sampled with a probability proportional to the number of instances in a given micro-cluster. The corresponding seed represents the centroid of that micro-cluster. In later iterations, the centre mc_i is the weighted centroid of the micro-cluster MC_i.

- Step 1: For each acquired instance X^t, MCOD finds (1) the nearest micro-cluster MC_i, whose centre mc_i is the nearest to it, and (2) the set of micro-cluster(s) R, whose centres are within a distance $\frac{3\times r}{2}$ from X^t.
- Step 2: If $dist(X^t, mc_i) \leq \frac{r}{2}$; such that mc_i is the centre of the nearest cluster MC_i, X^t is assigned to the corresponding micro-cluster.
- Step 3: Otherwise, a range query q for X^t is evaluated with respect to the instances in (1) the set PD and (2) the micro-clusters of the set R.

- Step 4: Consider n; the number of neighbours $N^{t'} \in PD$ to X^t, such that $dist(X^t, N^{t'}) \geq \frac{r}{2}$. If $n > \theta k; \theta \geq 1$, then a new micro-cluster with centre X^t is created and the n neighbours are assigned to this micro-cluster.
- Step 5: A micro-cluster whose size decreases below $k + 1$ is deleted and a range query similar to that described for X^t is performed for each of its former instances.
- Step 6: An instance X^t is flagged as an outlier, if there exists less than k instances in either PD or R.

7.2.2.2 Anytime Outlier Detection

Anytime Outlier detection (AnyOut) is a cluster-based technique that utilises the structure of ClusTree [134], an extension to R-tree [135], [136], to compute an outlier score. The tree structure of AnyOut suggests a hierarchy of clusters, such that the clusters at the upper level subsume the fine grained information at the lower level. ClusTree is traversed in top-down manner to compute the outlier score of an acquired instance X^t until it is interrupted by the subsequent (next) instance X^t. Thus, the descent down the tree improves the certainty of the outlier score, nevertheless, the later the arrival of the subsequent instance the higher the certainty [120].

Given that, a cluster is represented by a Cluster Feature tuple $CF = (n, LS, SS)$, such that n is the number of instances in the cluster, LS and SS are respectively the linear sum and the squared sum of these instances. The compact structure of the tree using CF tuples reduces space requirements. From BIRCH [121] to CluStream [122], cluster features and variations have been successfully used for online summarisation of data, with a possible subsequent offline process for global clustering.

Let e_s represent a cluster node entry in ClusTree. Given the defined two scores to compute the degree of outlierness of an instance X^t: (1) a mean outlier score $s_m(X^t)$=$dist(X^t, \mu(e_s))$, such that $\mu(e_s)$ is the mean of the cluster node entry $e_s \in$ ClusTree where X^t is interrupted; and (2) a density outlier score is $s_d(X^t)$=$1 - g(X^t, e_s)$, such that $g(X^t, e_s)$ is the Gaussian probability density of X^t for $\mu(e_s)$ as defined in [120], below we give a brief formalisation of AnyOut.

In the case of a constant data stream:

- Step 1: Initialisation: Each X^t in the data stream is assigned with an actual confidence value $conf(X^t)$=$e^{-s(X^t)}$.
- Step 2: Distribute the computation time for each X^t based on the scattered confidences.
- Step 3: For each acquired instance X^t, AnyOut traverses the tree in a top-down manner until the arrival of the instance X^{t+1} in the data stream.
- Step 4: At the moment of interruption, X^t is inserted to the cluster node entry $e_s \in$ ClusTree, where X^t arrives (pauses).
- Step 5: The outlier score of X^t, according to the specified parameter $s_m(X^t)$ or $s_d(X^t)$, is computed with respect cluster node entry e_s.
- Step 6: The expected confidence value for the outlier score of X^t is updated based on the computation time. *The confidence value (certainty) increases as the computation time increases.*

7.3 E-RAIDS: Ensemble of Random Subspace Anomaly Detectors In Data Streams

This Section presents the proposed *E-RAIDS* for insider threat detection with a detailed description of the feature subspace anomaly detection and the aggregate component (ensemble voting).

The established continuous distance-based outlier detection techniques (MCOD and AnyOut), described and formalised in Section 7.3.2, are utilised as building block outlier detection techniques techniques for the proposed E-RAIDS approach.

In this work, we propose a streaming anomaly detection approach, namely Ensemble of Random subspace Anomaly detectors In Data Streams (E-RAIDS), for insider threat detection. In other words, E-RAIDS is an ensemble of an established distance-based outlier detection technique (MCOD or AnyOut), such that each model of the ensemble learns on a random feature subspace. The idea of E-RAIDS is to employ an outlier detection technique on a feature subspace, to detect *local outliers* that might not be detected over the whole feature space. These local outliers may

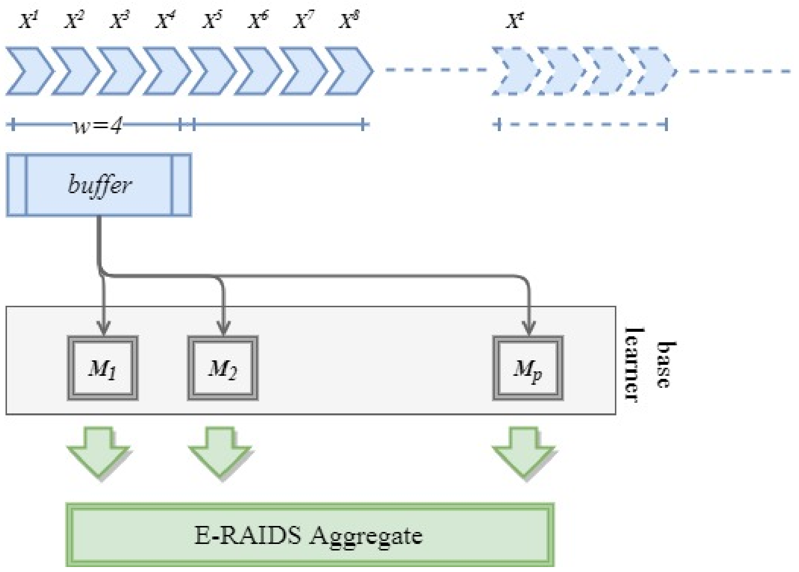


FIGURE 7.2: E-RAIDS framework. The set of blue arrows represent a data stream, where each arrow represents an instance X^t acquired at a session slot t. The buffer is a temporary short memory of allocated capacity (equals to w) which accumulates the instances in a data stream. $\{M_1, M_2, \ldots, M_p\}$ represents the p models of the ensemble which learn on p feature subspaces. The E-RAIDS aggregate component aggregates the results from the p feature subspaces to confirm whether to generate an alarm.

refer to anomalous behaviours (instances) attributed to a malicious insider threat. Hence, the ultimate aim of the E-RAIDS approach is to detect any-behaviour-all-threat (*threat hunting*); a process that leads to a reduction of the number of false alarms.

Fig. 7.2 presents the E-RAIDS framework. The set of blue arrows represent a data stream, where each arrow represents an instance (feature vector) X^t acquired at a session slot t. Consider the formalisations below:

- window: a segment of a fixed size w that slides along the instances in a data stream with respect to time (i.e. session slots);
- *buffer*: a temporary short memory of allocated capacity (equals to w). It temporarily accumulates the instances in a data stream. The *buffer* starts to accumulate the instances in a data stream at the start of the window and stops once the *buffer* is full after w number of instances. The full *buffer* is then input to the base learner component to be processed; and

- window iteration $wIter$: an iteration of the window slide. It starts at the already processed instances (in the previous *buffer*) $procInst$ and ends at $procInst$+w. For instance, $wIter$=0 starts at $procInst$=0 and ends at w. $wIter$=1 starts at $procInst$=w and ends at $2 \times w$. The window iteration $wIter$ allows the synchronisation between the window slide and the *buffer* accumulation.

Based on the aforementioned formalisation, at each window iteration $wIter$, the *buffer* accumulates w number of instances. Once the *buffer* is full, the instances in the *buffer* are input to the base learner component. A base learner component refers to the distance-based outlier detection technique to be utilised (MCOD or AnyOut). It employs a p number of base models to learn on randomly selected p feature subspaces. A feature subspace $FS_i \subseteq F$ is defined as a subset of features selected from the whole feature space f, where $F = \{f_1, f_2, \ldots, f_m\}$. The rationale behind the idea of random feature subspaces is to detect local outlier(s) (anomalous behaviour(s)) that might not be detected over the whole feature space.

Let p=$m+1$ represent the number of feature subspaces selected randomly, such that m is set to the number of features (dimensions) of the feature space F in the community data set. m represents the number of feature pairs (i.e. 2 features per subspace), and 1 represents the whole feature space (i.e. all features). The p feature subspaces are utilised to build the ensemble of p models $\{M_1, M_2, \ldots, M_m, M_p\}$, such that $\{M_1, M_2, \ldots, M_m\}$ learn on feature pairs, and M_p learns on the whole feature space. In this way, E-RAIDS is capable of detecting *local outliers* in the m feature pairs, as well as *global outliers* in the whole feature space.

7.3.1 Feature Subspace Anomaly Detection

For each model M_i over a feature subspace FS_i, we define the following data repositories and a survival factor:

- $outSet$: a temporary set of the outliers detected by a base learner (MCOD or AnyOut) at $wIter$;
- $outTempList$: a list that stores the triples $\langle out, wIter, tempC \rangle$: (1) an outlier $out \in outSet$, (2) the $wIter$ where it was first detected, and (3) a temporal count

$tempC$ that counts the number of subsequent windows where the outlier out was detected;

- $subVoteList$: a list that stores the triples $\langle wIter, subVote, subOutSet\rangle$: (1) a $wIter$, (2) a $subVote$ parameter set to 1 if the feature FS_i votes for an alarm to be generated at $wIter$ and 0 otherwise, and (3) an outlier set $subOutSet$; and
- vf_s: a survival factor that confirms whether a feature **s**ubspace FS_i votes for an alarm to be generated at $wIter$. In other words, if an outlier out survived (has been detected) for a vf_s number of subsequent windows, then out is defined as a *persistent* outlier. A persistent outlier boosts the chance of an alarm to be generated at $wIter$.

Over each feature subspace FS_i, the base learner (MCOD or AnyOut) processes the *buffer* at the current $wIter$ to update the model M_i. M_i identifies the outlier set $outSet$ that includes (1) the outliers from the *buffer* at the current $wIter$; and (2) the outliers from the previously learned instances before the model being updated. The type (2) outlier refers to an instance that was identified as an inlier, however, turned into an outlier in the current $wIter$.

For each outlier $out \in outSet$, if out does not exist in $outTempList$, a new triple $\langle out, wIter, 1\rangle$ is appended to $outTempList$. In this case, $tempC$ is assigned to 1 to declare that it is the first time an outlier out detected. If out exists in the $outTempList$, $tempC$ is incremented by 1 in the triple for out.

For each outlier $out \in outTempList$, E-RAIDS checks (1) if $tempC$=vf_s, then out survived for a vf_s number of subsequent windows. We call it a *persistent outlier*. Thus, a persistent outlier confirms that the FS_i votes for an alarm at $wIter$. Thus, $subVote$ is set to 1 in the triple for $wIter$ in $subVoteList$. (2) If $wIter - tempC$=vf_s, then out has turned into an inlier. We call it an *expired outlier*. Thus, the triple for the expired outlier out is removed from the $outTempList$. (3) If $wIter - tempC < vf_s$, then out is neither a persistent outlier nor an expired outlier. We call it a *potential outlier*.

Eventually, $outTempList$ consists of persistent outliers and potential outliers (expired outliers has been removed). E-RAIDS appends persistent outliers and potential outliers in $outTempList$ to the $subOutSet$ in the triple for $wIter$ in $subVoteList$.

Algorithm 3 Feature Subspace

```
1: $wIter \leftarrow 0$
2: foreach $X^t \in$ stream do
3:     accumulate $X^t$ to buffer
4:     if buffer is full then
5:         $outSet \leftarrow$ get outliers detected by base
6:         foreach $out \in outSet$ do
7:             if $outinoutTempList$ then
8:                 a new triple $\langle out, wIter, 1\rangle$ is appended to $outTempList$
9:             else
10:                increment $tempC$ by 1 for $out$ in $outTempList$
11:            end if
12:        end for
13:        foreach $out \in outTempList$ do
14:            if $tempC = vf_s$ then
15:                set $subVote$ to 1 in $subVoteList$ for the current $wIter$
16:                remove out from $outTempList$
17:                $subOutSet \leftarrow$ persistent $out\cup$ potential outliers
18:                append $subOutSet$ to $subVoteList$ at $wIter$
19:            end if
20:            if $wIter - tempC = vf_s$ then
21:                remove $out$ from $outTempList$
22:            end if
23:        end for
24:        increment $wIter$
25:    end if
26:    empty buffer
27: end for
28: return $subVoteList$
```

Finally, the *buffer* is emptied to be prepared for the subsequent (next) window of upcoming instances ($wIter + 1$). A step-by-step pseudo code for the E-RAIDS base learner procedure is provided in Algorithm 3.

Algorithm 4 Ensemble of Random Feature Subspaces

```
1: foreach $wIter$ do
2:     foreach $FS_i \in FS$ do
3:         $subVote \leftarrow$ get $subVote$ for $wIter$ from $subVoteList$ for $FS_i$
4:         $eVote \leftarrow eVote + subVote$ for $wIter$ in $eVoteList$
5:         $subOutSet \leftarrow$ get $subOutSet$ for $wIter$ from $subVoteList$ for $FS_i$
6:         append $subOutSet$ to $eOutSet$ for $wIter$ in $eVoteList$
7:     end for
8:     if $eVote = vf_e$ then
9:         flag an alarm
10:    end if
11: end for
```

7.3.2 Ensemble of Random Feature Subspaces Voting

For the ensemble of p models $\{M_1, M_2, \ldots, M_m, M_p\}$ on feature subspaces $\{FS_1, FS_2, \ldots, FS_m, FS_p\}$ respectively, we define the following data repository and a vote factor:

- $eVoteList$: a list that stores the triples (1) a $wIter$, (2) an $eVote$ parameter that counts the number of feature subspaces that vote for an alarm, and (3) an outlier set $eOutSet$ that appends the $subOutSet$ from each FS_i votes for an alarm. It has the form $\langle wIter, eVote, eOutSet\rangle$.

- vf_e: a vote factor that confirms whether an alarm to be generated at $wIter$ by the whole **e**nsemble. In other words, if a vf_e number of feature subspaces vote for an alarm, then an alarm is generated at $wIter$.

As illustrated in Fig. 7.2, the E-RAIDS aggregate component aggregates the results from the p feature subspaces, in order to confirm whether to generate an alarm or not at each window iteration $wIter$. For each feature subspace FS_i, if $subVote$=1 for $wIter$, E-RAIDS adds $subVote$ to $eVote$ for $wIter$ in $eVoteList$. Furthermore, E-RAIDS gets $subOutSet$ for $wIter$ from $subVoteList$, and appends to $eOutSet$ for $wIter$ in $eVoteList$.

After getting the votes from all the feature subspaces in the ensemble, E-RAIDS checks if $eVote$=vf_e. If the condition is satisfied, then an alarm of a malicious insider threat is generated at $wIter$. The voting mechanism is technically controlled by the vote factor vf_e, such that if a vf_e number of feature subspaces vote for anomalous behaviour(s) at a window iteration $wIter$, then an alarm is generated. This accordingly handles the case of a conflict, where $\frac{p}{2}$ (50%) of the feature subspaces in the ensemble vote for anomalous behaviour(s) and the other $\frac{p}{2}$ vote for normal behaviour(s). The ensemble technically checks if $\frac{p}{2} = vf_e$, then an alarm is generated.

A step-by-step pseudo code for the E-RAIDS aggregate procedure is provided in Algorithm 4.

7.4 Experiments

We evaluated the effectiveness of the proposed approach on the CMU-CERT data sets using Windows Server 2016 on Microsoft Azure (RAM $140GB$, OS $64-bits$, CPU Intel Xeon $E5-2673v3$) for data pre-processing and Microsoft Windows 7 Enterprise (RAM $12GB$, OS $64-bits$, CPU Intel Core $i5-4210U$) for experiments. We implemented E-RAIDS-MCOD and E-RAIDS-AnyOut and carried out the experiments in Java environment (Eclipse Java Mars) using the open-source **MOA** package [133].

7.4.1 Experimental Tuning

In this work, we define two experiments based on the proposed E-RAIDS approach. The E-RAIDS-MCOD learns an ensemble of p MCOD base learners, such that each MCOD base learner is trained on a feature subspace of the p feature subspaces. Likewise, the E-RAIDS-AnyOut learns an ensemble of p AnyOut base learners.

The experiments are tuned for different values of parameters. Table 7.1 presents the values for the parameters tuned in each of MCOD and AnyOut, with a short description. *Note that an extensive number of experiments was done to select the presented tuning values for the parameters. The values were selected based on E-RAIDS achieving the best performance in terms of the evaluation measures described below.*

For MCOD, the parameter k has a default value k=50 in the MOA package. In this Chapter, we present k=$\{50, 60, 70\}$ to evaluate the E-RAIDS-MCOD for different number of neighbours. The parameter r, with a default value r=0.1, is presented for a range for r=$\{0.3, 0.4, 0.5, 0.6, 0.7\}$ to check the influence of r.

The range of values presented for the parameters k and r provided the best results on the CMU-CERT data sets in terms of TP$_T$ and FPAlarm.

For AnyOut, the parameter $|D_{train}|$ is presented for 500 instances, knowing that no threats are present at the beginning of the stream in these 500 instances. The parameters $confAgr$ and $conf$ did not show a significant influence on the utilised data sets, so both are assigned to their default values, in the MOA package, 2 and 4 respectively. τ has a minimum value 0 and a maximum value 1, so it is presented for τ=$\{0.1, 0.4, 0.7\}$ to evaluate the influence of varying the outlier score threshold

TABLE 7.1: Tuned Parameters.

MCOD parameter	description
k={50, 60, 70}	number of neighbours parameter
r={0.3, 0.4, 0.5, 0.6, 0.7}	distance parameter for outlier detection
AnyOut parameter	**description**
$\|D_{train}\|$=500	training set size
$confAgr$=2	size of confidence aggregate
$conf$=4	initial confidence value
τ={0.1, 0.4, 0.7}	outlier score threshold
$oscAgr$={2, 4, 6, 8}	size of outlier score aggregate

on the outliers detected. $oscAgr$ has a minimum value 1 and a maximum value 10, so it is presented for the $oscAgr$=$\{2, 4, 6, 8\}$.

The range of values for the parameters τ and $oscAgr$ are presented to analyse the effect of varying the values on the performance of E-RAIDS-AnyOut.

In general, for E-RAIDS approach with either MCOD or AnyOut base learner, we present the vouch factor vf_s=2, so it confirms an outlier as positive (anomalous) after it survives for 2 subsequent windows. We present the vote factor vf_e=1, so if at least 1 feature subspace in the ensemble flags an alert, an alarm of a malicious insider threat is confirmed to be generated.

The window size w, which is equal to the capacity of the buffer, is a main parameter to be tuned in the proposed E-RAIDS. For both E-RAIDS-MCOD and E-RAIDS-AnyOut, the window size is presented for $w = \{50, 100, 150, 200\}$. The presented range of values for the parameter w provided the best performance on the CMU-CERT data sets in terms of detecting malicious insider threats, and reducing false alarms.

As previously described, an instance (feature vector) is a set of events executed during a pre-defined session slot. In this work, we define a session slot per 4 hours. Let w represent the window size that accumulates the acquired instances in a data stream. If w=50 and vf_s=2, then the instances in each window are processed after $4 \times$ 50=200 hours $\simeq$ 8 days. Based on the description of E-RAIDS, a threat (outlier) may be confirmed as an outlier at least over the window it belongs to (after 8 days) or over the next window (after $\simeq 8 \times 2$=16 days). If w=200 and vf_s=2, then the instances in each window are processed after $4 \times$ 200=800 hours $\simeq$ 33 days. In other words, a threat (outlier) may be confirmed as an outlier at least after 33 days or over the next window (after $\simeq 33 \times 2$=66 days). Note that the malicious insider threats simulated in the CMU-CERT data sets span over at least one month and up to 4 months. Hence, we set the following targets of E-RAIDS in Target 1.

Objective 1: *The E-RAIDS approach is capable of detecting the malicious insider threats in real-time (during the time span of the undergoing threat).*

7.4.2 Evaluation Measures

As aforementioned, the ultimate aim of the E-RAIDS approach is to detect all the malicious insider threats over a data stream in real-time, while reducing the number of false alarms.

The challenge of the insider threat problem lies in the variety and complexity of the malicious insider threats in the data sets. Each malicious insider threat comprises a set of anomalous behaviours. An anomalous behaviour is represented by an instance (feature vector) that describes a set of events (features) carried out by a malicious insider. Based on the challenge of the problem, we formulate the E-RAIDS approach with the aim to detect *any-behaviour-all-threat*. This means that it is sufficient to detect any anomalous behaviour (instance) in all malicious insider threats. Hence, the E-RAIDS approach is formulated as a *threat hunting* approach, where a threat is detected if (1) a vf_e number of feature subspaces confirm an undergoing threat (flag alarm) over a window and (2) the outliers, associated to the alarm flagged over the window, include at least one True Positive (anomalous behaviour detected as outlier). Note that although the detection of one behaviour confirms detection of the threat, we set the following target of E-RAIDS in Target 2.

Objective 2: *The E-RAIDS approach is capable of detecting more than one behaviour from the set of behaviours that belong to a malicious insider threat. We refer to as (more than one)-behaviour-all-threat detection in E-RAIDS.*

Furthermore, the property of the E-RAIDS approach of using windows over data streams introduces a refined version of the evaluation of False Positives (FP). In this work, we use the FPAlarm to evaluate the false positives. If all the outliers associated with the alarm generated over a window are actually normal, then the alarm is considered a False Alarm (i.e. FPAlarm is incremented by 1). A formal definition for FPAlarm is given in the following.

Unlike the evaluation measures utilised in Chapter 5 and Chapter 6, in this Chapter, we define different measures used to evaluate the performance of E-RAIDS. The rationale behind this is the streaming nature of the E-RAIDS approach. The evaluation measures include a refined version of the default (known) evaluation measures, taking our ultimate aim into account.

- **TP**$_T$: *True Positives* a refined version of the default TP to evaluate the number of threats detected by the framework among all the P$_T$ malicious insider threats. TP$_T$ is incremented if at least one anomalous instance (behaviour attributed to the threat) is associated as an outlier with a flagged alarm;
- **FPAlarm**: *False Positive Alarm* a refined version of the default FP to evaluate the number of false alarms generated. An alarm is declared false if all the outliers associated with the alarm are actually normal instances. This means that none of the instances contributed to generating the alarm, therefore, it is a false alarm;
- FN$_T$: *False Negatives* a refined version of the default FN to evaluate the number of insider threats not detected; and
- **F1** measure: defined based on the values of the above defined measures.

The evaluation for E-RAIDS includes not only the defined evaluation measures, but also proving the previously stated targets.

7.4.3 Results and Discussion

In this work, the results are presented and discussed for E-RAIDS-MCOD and E-RAIDS-AnyOut as follows: in terms of (1) the pre-defined evaluation measures; (2) voting feature subspaces; (3) real time anomaly detection; and (4) the detection of (more than one)-behaviour-all-threat.

TABLE 7.2: Maximum TP_T of detected insider threats over communities associated with (1) the number of neighbours k, the distance parameter r, and the window size w which achieved the maximum TP_T for E-RAIDS-MCOD, and (2) the outlier score threshold τ, the size of outlier score aggregate $oscAgr$, and the window size w which achieved the maximum TP_T for E-RAIDS-AnyOut. Note that the parameter combination(s) which achieved the maximum TP_T are listed. The parameters are displayed as $\langle k, r, w\rangle$ for E-RAIDS-MCOD and $\langle \tau, oscAgr, w\rangle$ for E-RAIDS-AnyOut.

community	E-RAIDS-MCOD	Parameters k, r, w
com-P	**16**	50,0.3,100 60,0.4,50
		60,0.6,100 60,0.7,150
		70,0.4,50
com-S	**21**	70,0.4,150
com-I	10	50,0.4,50 70,0.3,100
community	**E-RAIDS-AnyOut**	**Parameters $\tau, oscAgr, w$**
com-P	**16**	0.1,2,100-200 $\{0.3, 0.7\}$,2,50-200
com-S	20	0.1,2,50-100 0.3,2,50-100
		0.7,2,150
com-I	**12**	0.1,2,50-200 0.3,2,50-100
		0.7,2,$\{50, 200\}$

7.4.3.1 MCOD versus AnyOut Base Learner for E-RAIDS in Terms of Evaluation Measures

In the following, we analyse the performance of E-RAIDS with MCOD base learner versus AnyOut base learner in terms of the pre-defined evaluation measures: TP_T out of P_T; FPAlarm; and F1 measure.

Tables 7.2 and 7.3 present the maximum TP_T and the minimum FPAlarm attained by E-RAIDS-MCOD and E-RAIDS-AnyOut over the communities. The results are reported in terms of the parameter values in the given sequence k, r, w for E-RAIDS-MCOD and $\tau, oscAgr, w$ for E-RAIDS-AnyOut respectively.

TABLE 7.3: Minimum FPAlarm over communities associated with (1) the number of neighbours k, the distance parameter r, and the window size w which achieved the minimum FPAlarm for E-RAIDS-MCOD, and (2) the outlier score threshold τ, the size of outlier score aggregate $oscAgr$, and the window size w which achieved the minimum FPAlarm for E-RAIDS-AnyOut. Note that the parameter combination(s) which achieved the minimum FPAlarm are listed. The results are displayed as $\langle k, r, w\rangle$ for E-RAIDS-MCOD and $\langle \tau, oscAgr, w\rangle$ for E-RAIDS-AnyOut.

community	E-RAIDS-MCOD	Parameters k, r, w
com-P	**0**	50,0.4,200 60,$\{0.3, 0.5, 0.6\}$,200
com-S	**0**	50,0.4,200 50,0.7,100
		60,0.3,200 60,0.5-0.7,100
		70,0.6-0.7,150
com-I	**0**	50,0.3,200 60,0.5,200
		70,0.5-0.7,200
community	**E-RAIDS-AnyOut**	**Parameters $\tau, oscAgr, w$**
com-P	2	$\forall\tau$,$\forall oscAgr$,200
com-S	2	$\forall\tau$,$\forall oscAgr$,150-200
com-I	1	0.1,$\{2, 4\}$,200 0.3,2-6,200
		0.7,2-4,200

E-RAIDS-MCOD Fig. 7.3 presents the variation of $F1$ measure as a function of window size w for E-RAIDS with MCOD base learner over the communities. The results are reported with respect to k and r parameter values.

A preliminary analysis of the F1 measure shows no evident pattern in terms of any of the parameters k, r, or w. Over the community com-P, E-RAIDS-MCOD achieves the maximum F1=0.9411; k=60, r=0.7, w=150. It detects the maximum TP$_T$=16 out of P$_T$=17, thus missing one malicious insider threat. However, it flags only one false positive alarm (FPAlarm=1). Furthermore, Table 7.3 shows that E-RAIDS-MCOD reports the minimum FPAlarm=0 at w=200 for different values of k and r.

Over the community com-S, E-RAIDS-MCOD achieves the maximum F1=0.9523; k=70, r=$\{0.6, 0.7\}$, w=150. It detects a TP$_T$=20 out of P$_T$=22, while flagging no false positive alarms (FPAlarm= 0). The maximum TP$_T$=21 is attained at k=70, r=0.4, w=150, however, flagging FPAlarm=2.

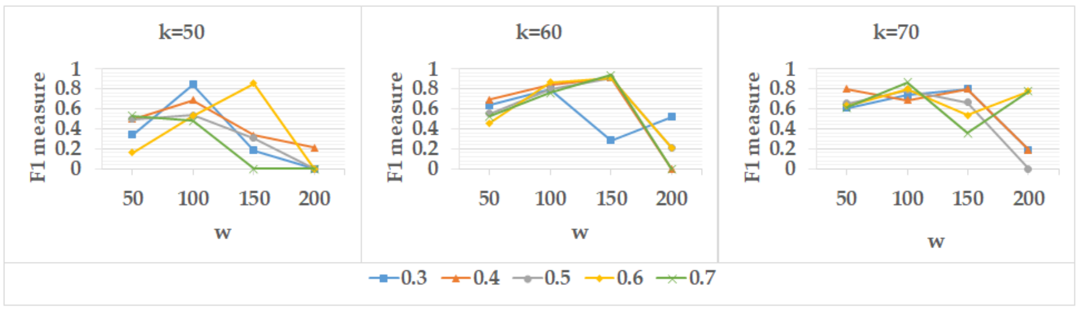
Over the community com-I, E-RAIDS-MCOD achieves the maximum F1=0.6451; k=70, r=0.3, w=100. It detects a TP$_T$=10 out of P$_T$=12, thus missing two malicious insider threats, while flagging FPAlarm=9. Nevertheless, Table 7.3 shows that E-RAIDS-MCOD reports the minimum FPAlarm=0 at w=200 for different values of k and r.

We can deduce that, in these experiments, the window size w=$150, 200$ give the best performance for E-RAIDS-MCOD in terms of the evaluation measures.

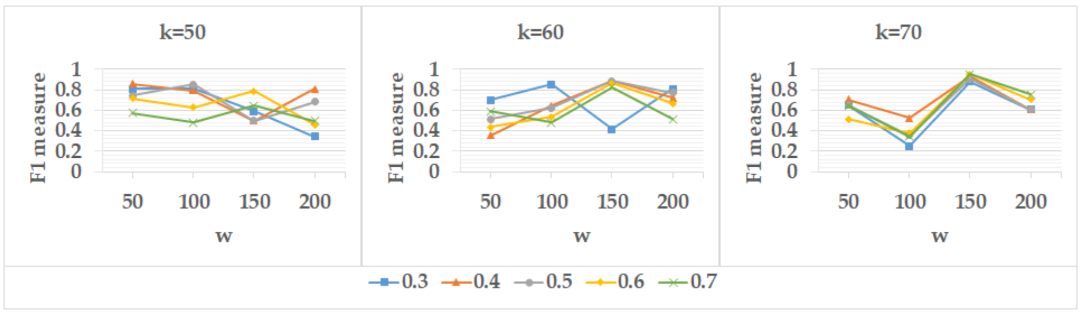
E-RAIDS-AnyOut Fig. 7.4 presents the variation of F1 measure as a function of window size w for E-RAIDS with AnyOut base learner over the communities. The results are reported with respect to τ and $oscAgr$ parameter values.

It is evident that F1 measure is inversely proportional to the parameter $oscAgr$ for E-RAIDS-AnyOut. The values of F1 measure at $oscAgr$=2 is the highest with respect to all the window sizes w=50 to 200 over all communities. Moreover, Figure 7.4 reveals that F1 measure is directly proportional to the parameter w. The value of F1 measure increases as the window size w increases.

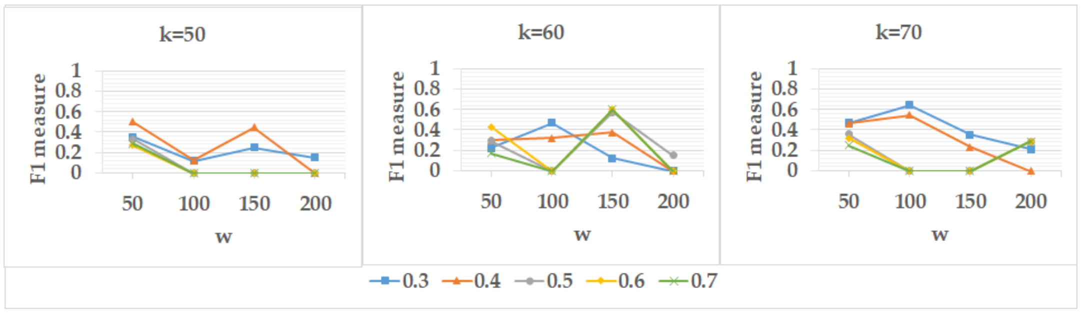
Over the community com-P, E-RAIDS-AnyOut achieves the maximum F1=0.9142; $\forall\tau$, $oscAgr$=2, w=200. It detects the maximum TP$_T$=16 out of P$_T$=17, while reducing



(A) com-P.



(B) com-S.



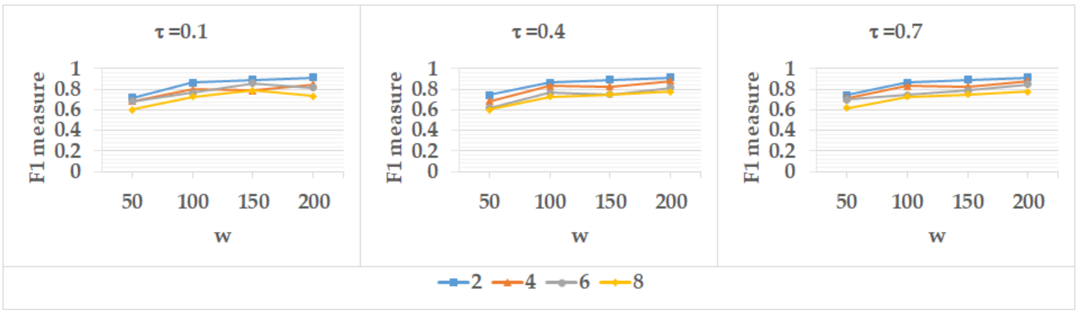
(C) com-I.

FIGURE 7.3: The variation of F1 measure as a function of window size w for E-RAIDS with MCOD base learner over the communities. The x-axis represents the window size $w = \{50, 100, 150, 200\}$, the y-axis represents the values of F1 measure, and the legend represents the values of the radius parameter r=$\{0.3, 0.4, 0.5, 0.6, 0.7\}$. The graphs in Fig. 7.3a, Fig. 7.3b, and Fig. 7.3c represent the results for k=$\{50, 60, 70\}$.

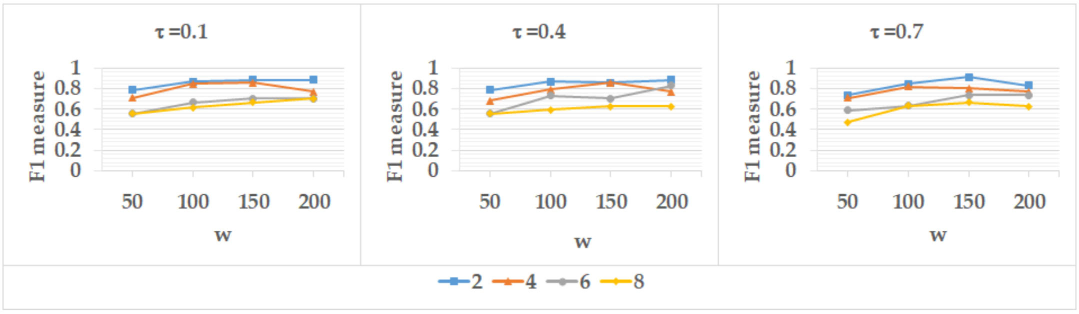
the false positive alarms to FPAlarm=2. Table 7.3 shows that E-RAIDS-AnyOut reports the minimum FPAlarm=2 $\forall\tau, \forall oscAgr$ at w=200. Thus, the higher the window size, the lower the FPAlarm. Table 7.2 shows that E-RAIDS-AnyOut reports the maximum TP$_T$=16 at $oscAgr$=2 in general terms $\forall\tau, \forall w$. Thus, the lower the $oscAgr$, the higher the TP$_T$ detected.

Over the community com-S, E-RAIDS-AnyOut achieves the maximum F1=0.9090; τ=0.7, $oscAgr$=2, w=150. It detects a TP$_T$=20 out of P$_T$=22, while flagging two false positive alarms (FPAlarm=2).

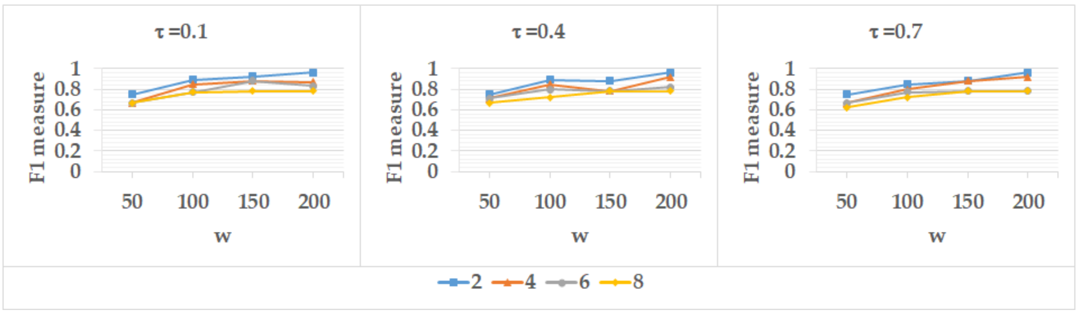
Over the community com-I, E-RAIDS-AnyOut achieves the maximum F1=0.9600;



(A) com-P.



(B) com-S.



(C) com-I.

FIGURE 7.4: The variation of F1 measure as a function of window size w for E-RAIDS with AnyOut base learner over the communities. The legend represents the $oscAgr$ parameter values. The x-axis represents the window size $w = \{50, 100, 150, 200\}$, the y-axis represents the values of F1 measure, and the legend represents the values of $oscAgr$=$\{2, 4, 6, 8\}$. The graphs in Fig. 7.4a, Fig. 7.4b, and Fig. 7.4c represent the results for τ=$\{0.1, 0.4, 0.7\}$.

$\forall\tau$, $oscAgr$=2, w=200. It detects a TP$_T$=12 out of P$_T$=12, while flagging one false positive alarm (FPAlarm=1).

In terms of the evaluation measures, E-RAIDS-MCOD outperforms E-RAIDS-AnyOut over the communities, where E-RAIDS-MCOD achieves a higher F1 measure over com-P and com-S, a higher TP$_T$ over com-S, and a lower FPAlarm=0 over all communities.

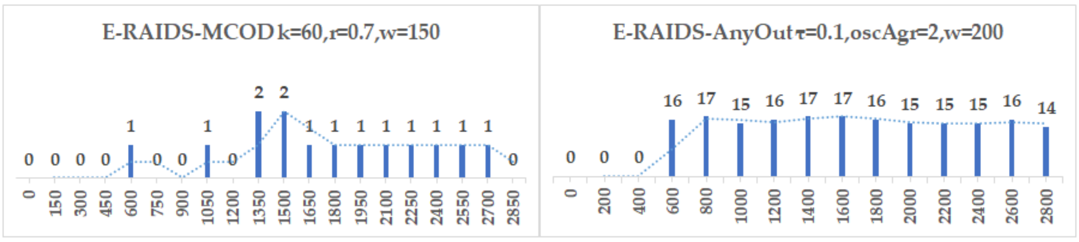
7.4.3.2 MCOD versus AnyOut for E-RAIDS in terms of Voting Feature Subspaces

In the following, we address the merit of using feature subspaces in the E-RAIDS approach. We compare the number of feature subspaces in the ensemble that voted for a malicious insider threat in each of E-RAIDS-MCOD and E-RAIDS-AnyOut. The rationale behind using a random feature subspace to train each model of the p models in the ensemble, is to train the base learner (MCOD or AnyOut) on a subset of the features and to detect local outliers that might not be detected over the whole feature space.

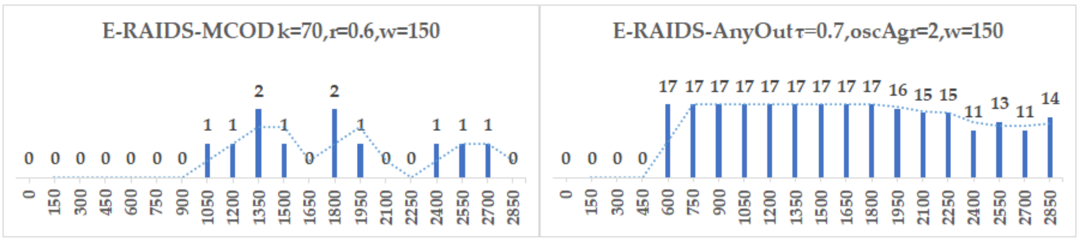
Figure 7.5 illustrates the number of votes that contributed in flagging an alarm of a malicious threat with respect to the number of instances processed (given the window size w) over the communities. The number of votes actually corresponds to the number of feature subspaces in the ensemble that generated an alert. We selected E-RAIDS-MCOD and E-RAIDS-AnyOut with their parameters that showed the best performance in terms of the evaluation measures. For com-P, Figure 7.5a shows the E-RAIDS-MCOD at k=60, r=0.7, w=150 and E-RAIDS-AnyOut at τ=0.1, $oscAgr$=2, w=200. For com-S, Figure 7.5b shows the E-RAIDS-MCOD at k=70, r=0.6, w=150 and E-RAIDS-AnyOut at τ=0.7, $oscAgr$=2, w=150. For com-I, Figure 7.5c shows the E-RAIDS-MCOD at k=70, r=0.3, w=100 and E-RAIDS-AnyOut at τ=0.1, $oscAgr$=2, w=200. Given that the $|D_{train}|$=500, this justifies why the votes for E-RAIDS-AnyOut start after 500 training instances has been processed over all communities.

Given the window size w, a window iteration $wIter$ starts at the number of already processed instances $procInst$ and ends at $procInst$+w. For example, given w=100, the first window iteration $wIter$=0 starts at $procInst$=0 and ends at $procInst+w$=100; $wIter$=1 starts at 100 and ends at 200; etc.

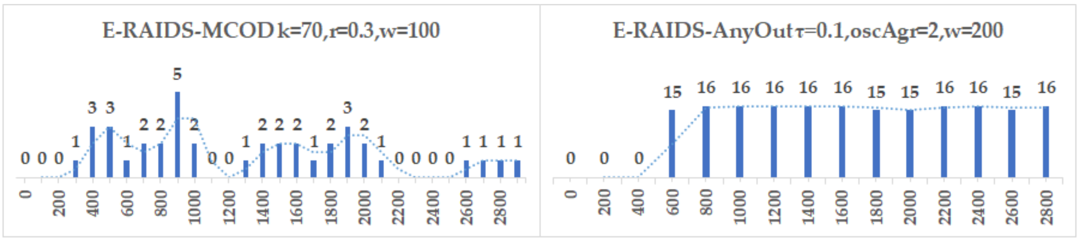
A preliminary analysis of the results shows that E-RAIDS-AnyOut flags alarms continuously for all $wIter$ after $procInst$=500. However, E-RAIDS-MCOD shows distinct alarms flagged, with no alarms at certain windows iterations. We recall that E-RAIDS-MCOD outperforms E-RAIDS-AnyOut in terms of FPAlarm measure. The continuous alarms flagged $\forall wIter$ in E-RAIDS-AnyOut explain the higher FPAlarm, as well as they reveal the uncertainty of E-RAIDS-AnyOut compared to E-RAIDS-MCOD.



(A) com-P.



(B) com-S.



(C) com-I.

FIGURE 7.5: The number of votes which contributed in flagging an alarm of a malicious threat with respect to the number of instances processed over the communities. The x-axis represents the number of instances processed, and the y-axis represents the number of votes.

Knowing that the number of feature subspaces utilised in the ensemble is $p = 17$, the number of votes $\forall wIter$ in E-RAIDS-AnyOut has a minimum $votes$=11 and a maximum $votes$=17, which reveals a level of uncertainty. The case where 17 feature subspaces vote for an alarm, indicates that all (17) models of the ensemble detect *at least one* outlier (positive) associated to a malicious insider threat. On the other hand, the number of votes in E-RAIDS-MCOD has a maximum $votes$=2 (3 or 5 in rare cases). This means that only 1 or 2 feature subspaces vote for an alarm. We recall the complexity of the malicious insider threat scenarios in the CMU-CERT data sets. The anomalous instances usually exist in (sparse or dense) regions of normal instances, and rarely as *global outliers* with respect to the whole feature space. To address this, the E-RAIDS approach aims to detect *local outliers* that may be found over ANY (not ALL) of feature subspaces. Having all the feature subspaces in E-RAIDS-AnyOut voting for a threat, compared to a couple (1 or 2) of feature subspaces in E-RAIDS-MCOD, reinforces the uncertainty of E-RAIDS-AnyOut. We argue that the reason behind the uncertain performance of E-RAIDS-AnyOut is due to the interruption event in AnyOut where the outlier score of an instance X^t is computed upon the arrival of a new instance X^{t+1}. In fact, the processing of the instance X^t is interrupted at a certain level of the ClusTree (refer to 7.2.2.2), and therefore the outlier score is computed with a lower level of confidence (i.e. uncertain). The issue of uncertainty (i.e. lower level of confidence associated to an outlier score) is not actually faced in E-RAIDS-MCOD, because the underlying MCOD technique does not implement an interruption event.

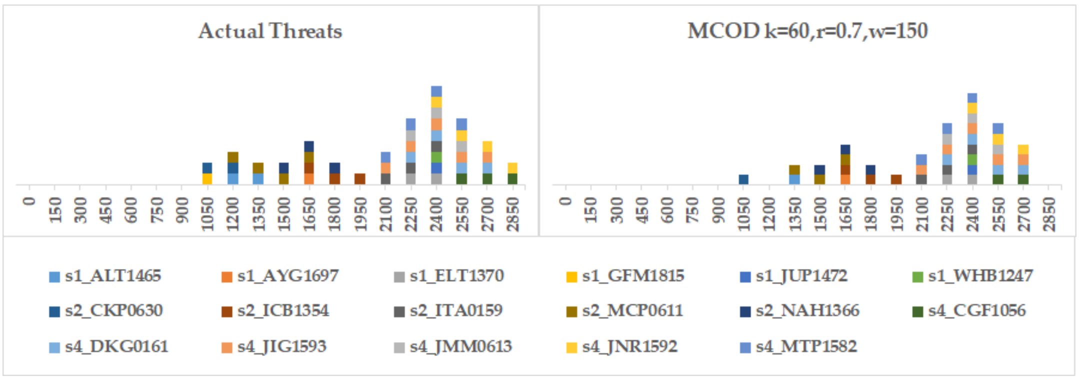
7.4.3.3 Real Time Anomaly Detection in E-RAIDS

To prove Target 1 to be true, it is necessary to check if the E-RAIDS detects the malicious insider threats in real-time (where real-time means that the alarm is flagged during the time span of the undergoing threat). Based on the previous conclusion regarding the uncertainty of E-RAIDS-AnyOut, and the superiority of E-RAIDS-MCOD in terms of (1) the evaluation measures and (2) the voting feature subspaces, we select E-RAIDS-MCOD to verify Target 1.

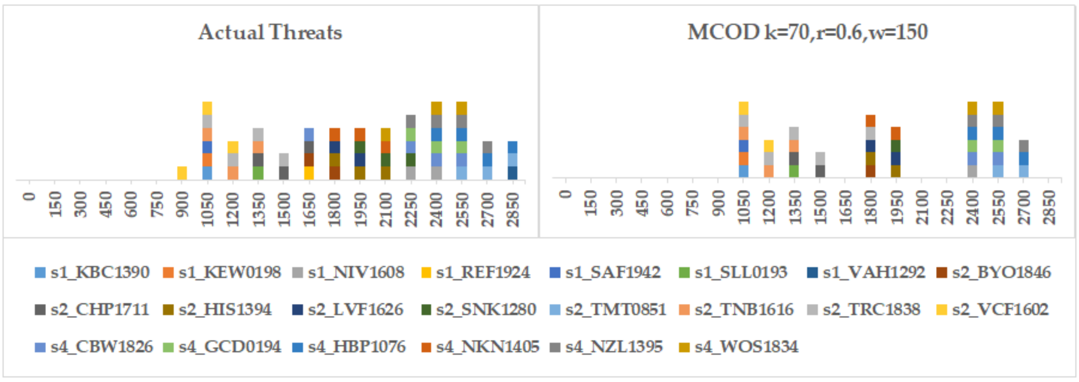
Fig. 7.6 illustrates the actual malicious insider threats versus the threats detected in E-RAIDS-MCOD with respect to the number of instances processed over the communities. The malicious insider threats are displayed in the legend over each community using the following label scenRef_insiderID (e.g. s1_ALT1465), such that scenRef (e.g. s1, s2, s3, or s4) represents the reference number for the scenario followed in the malicious insider threat; and insiderID (e.g. ALT1465, AYG1697) represents the user ID of the insider attributed to the threat. Hence, each colour in the legend refers to a malicious insider threat.

We observe in Fig. 7.6 that a malicious insider threat is detected either (obs1) at the current window iteration $wIter$ where it is actually simulated, or (obs2) at the subsequent (next) $wIter$. Hence, Target 1 is verified.

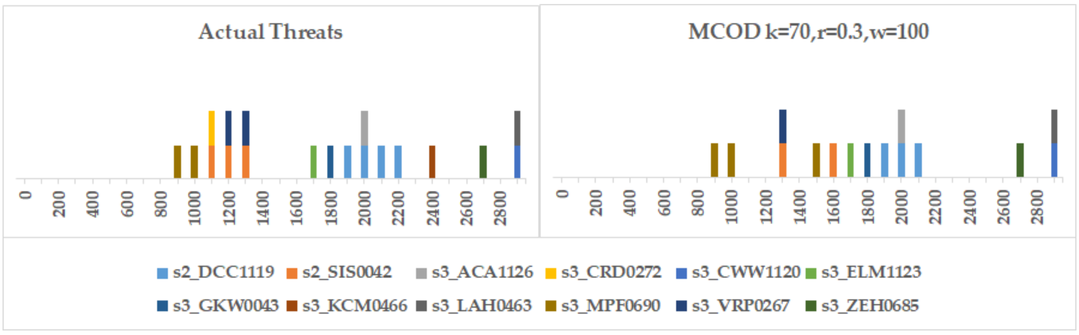
Based on the description of the E-RAIDS approach, a feature subspace votes for a threat at $wIter$ if at least one outlier survived for a vf_s number of subsequent windows, and consequently a specific threat is detected at $wIter$ if (cond1) a vf_e number of subspaces vote (an alarm is flagged), and (cond2) at least one outlier (positive) associated with the alarm belongs to the threat. However, the outliers associated with



(A) com-P.



(B) com-S.



(C) com-I.

FIGURE 7.6: The actual malicious insider threats versus the threats detected in E-RAIDS-MCOD with respect to the number of instances processed over the communities. The legend (colors) represents the malicious insider threats over each community using the following label scenRef_insiderID (e.g. s1_ALT1465). The x-axis represents the number of instances processed, and the y-axis presents the colors (malicious insider threats).

the alarm (as mentioned in (cond2)) consist of *persistent* outliers (that survived from $wIter$-1) and *potential* outliers (at the current $wIter$). A potential outlier, if satisfies (cond2), allows real-time detection at the current $wIter$ (obs1). A persistent outlier, if satisfies condition (cond2), allows real-time detection as observed at the subsequent $wIter$ (obs2).

7.4.3.4 (More than One)-Behaviour-All-Threat Detection in E-RAIDS

The final analysis addresses Target 2. The idea of *threat hunting* aims to detect any-behaviour-all-threat, however, Fig. 7.6 shows the capability of E-RAIDS-MCOD to detect more than one behaviour (not only one) from the set of behaviours that belong to a malicious insider threat. It manifests as multiple alarms (colour spikes) generated for a specific threat over a number of windows. Hence, Target 2 is verified.

Analytically, this manifests in multiple outliers associated with the alarm(s) flagged over window(s), which are actually true positives belonging to a specific malicious insider threat.

7.5 Summary

This Chapter addresses the shortcoming of high number of false alarms in the absence of labelled data (i.e. low data maturity). We present a streaming anomaly detection approach, namely Ensemble of Random subspace Anomaly detectors In Data Streams (E-RAIDS), for insider threat detection.

E-RAIDS is built on top of the established continuous outlier detection techniques (MCOD or AnyOut). These techniques use *data stream clustering* to optimise the detection of outliers (that may refer to malicious insider threats). E-RAIDS is an ensemble of p outlier detectors (p MCOD base learners or p AnyOut base learners), where each model of the p models learns on a random feature subspace. The merit of using feature subspaces is to detect local outliers that might not be detected over the whole feature space. These outliers may refer to anomalous behaviour(s) that belong to a malicious insider threat. Unlike E-RAIDS, the proposed framework in Chapter 6 presents an ensemble of k base anomaly detection models, where each model learns on a normal cluster.

To reduce the number of false alarms, E-RAIDS introduces two factors: as survival factor vf_s, and a vote factor vf_e. vf_s is defined to confirm whether a feature subspace votes for an alarm at each window iteration. vf_e is implemented in an aggregate component to control whether the aggregated votes over the ensemble confirm to generate an alarm.

We define two experiments: E-RAIDS-MCOD with MCOD base learner, and E-RAIDS-AnyOut with AnyOut base learner. We compare the performance of E-RAIDS using each of MCOD and AnyOut in terms of, (1) the evaluation measures: F1 measure, TP$_t$ of threats detected, and FPAlarm flagged; (2) the effectiveness of the concept of voting feature subspaces; (3) the capability of E-RAIDS to detect insider threats in real-time (Target 1); and (4) the capability of E-RAIDS to detect more than one behaviour belonging to an insider threat (Target 2) despite our formulation to the opportunistic approach.

The results show that E-RAIDS-MCOD outperforms E-RAIDS-AnyOut, where the latter shows a low level of certainty in the detection of outliers. E-RAIDS-MCOD reports a higher F1 measure=0.9411 and 0.9523 over com-P and com-S, no FPAlarm=0 over all communities, and misses only one threat TP$_T$=16 and 21 over com-P and com-S. Furthermore, E-RAIDS-MCOD demonstrates the capability of detecting more than one behaviour per threat in real-time.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

This research focuses on the detection of insider threats posed by malicious insiders in an organisation. The recent insider threat cases such as, Harold Martin, Edward Snowden, and Bradley manning (now known as Chelsea Manning), have increased the concern of academia, companies, and governments to develop effective insider threat detection mechanisms.

In this thesis, we address the shortcoming of the high number of false alarms (false positives). A False Positive – *FP* – refers to normal behaviour(s) detected or predicted as anomalous by a detection system. These FPs are often normal behaviours having a high similarity to neighbour anomalous behaviours. Analytically, this manifests as normal instances (feature vectors) whose feature values are similar to the feature values of anomalous instances. In a detection system, the aim to detect the malicious insider threats is substantial, nevertheless, the aim to mitigate the high number of false alarms is inevitable. The high number of false alarms flagged continuously by the detection system, deceive administrator(s) or analyst(s) about ongoing suspicious behaviours of many users who are actually benign. An administrator may spend a lot of time investigating the suspected users. Besides, the benign users accused of malicious intents would lose trust in their chief executives, which would reflect on their productivity and commitment to the organisation.

The ultimate aims of the proposed approaches were to detect any-behaviour-all-threat (Aim 1), while reducing the false alarms (Aim 2) as outlined in Section 1.4. To fulfil the aims, the objectives stated in Section 1.3 must be met.

In this research, *for the first time*, we formulated the machine learning approach to detect insider threats as an *opportunistic* approach (threat hunting) as defined in Section 3.3 in Chapter 3. The aim of an opportunistic approach is set to detect any-behaviour-all-threat; it is sufficient to detect any anomalous behaviour (instance) for all malicious insider threats in the data sets. In other words, we hunt a malicious insider threat by identifying at least one anomalous behaviour attributed to it, but not all behaviours. The design of the opportunistic approach with such a relaxing condition contributed to reducing frequent false alarms (attained objective 1).

We defined the feature space for the insider threat problem in Chapter 4. According to the literature, features in the established feature set were categorised into five groups, including frequency-based features, time-based features, Boolean features, attribute-based features, and others. Based on this feature set, we extracted feature vectors (instances) from activity logs to best describe the behaviour of users.

This work was conducted on synthetic insider threat data sets generated by CMU-CERT, where the CMU-CERT data is the only available data repository which implements malicious insider threat scenarios. We utilised the $r5.2$ data sets which consist of a considerable variety of malicious insider threats. We constructed community behaviour profiles, with session slots per four hours, for users having the same role in order to represent the correlation among users' behaviour in a community. The constructed community data sets are: Production line worker – com-P, Salesman – com-S, and IT admin – com-I. Recall that com-P consists of 2730 normal behaviours (instances), 366 anomalous behaviours (associated to 17 malicious insider threats), and an IR=7.4590; com-S consists of 2579 normal behaviours, 515 anomalous behaviours (associated to 22 malicious insider threats), and an IR=5.0077; and com-I consists of 2964 normal behaviours, 132 anomalous behaviours (associated to 12 malicious insider threats), and an IR=22.4545.

In this research, the insider threat problem was inspected from three different perspectives. In particular, we outline our three contribution approaches to detect insider threats based on the levels of data maturity as described in Section 1.4. Table 8.1 lists the levels of data maturity in an organisation; the objectives outlined to inspect the insider threat problem based on the data maturity; and a brief description of the approaches proposed to meet the objectives.

TABLE 8.1: Summary of research contribution approaches.

Data maturity	Objective	Chapter	Contribution Approach	Base Learner(s)	Keywords

high	Objective 2	Chapter 4	CD-AMOTRE	RF
				XGBoost
				SVM
					class imbalance data
					class decomposition
					minority oversampling
					trapper removal

medium	Objective 3	Chapter 6	E-base-OU	ocSVM iForest	one-class data class decomposition ensemble FP chunks progressive update FP oversampling
low	Objective 4	Chapter 7	E-RAIDS	MCOD AnyOut	stream data feature subspaces ensemble clustering outlier detection real-time

In the case of high data maturity, the insider threat problem is inspected as a class imbalance data problem, where there exists a rare occurrence of anomalous behaviours (minority class) among the normal baseline of users' behaviours (majority class). To meet objective 2, in Chapter 4, we proposed the *first* class imbalance approach for insider threat detection, namely CD-AMOTRE. To address the issue of class imbalance in data, we introduced a class decomposition concept, and a *novel* artificial minority oversampling technique (AMO-TRE) with a trapper removal procedure. In this approach, the class decomposition is applied to weaken the impact of the majority (normal) class. The normal class data is decomposed into k clusters, so that the two-class data ('normal' class label versus 'anomalous' class label), is transformed to a multi-class data (k 'normal' cluster (class) labels versus 'anomalous' class label(s)). Introducing the concept of class decomposition contributed into reducing the FPs, though the class data may not be perfectly clustered. The role of the AMOTRE technique is to oversample the minority (anomalous) class data to generate artificial samples of the anomalous class. The merit of the oversampling procedure is that it is selective. AMOTRE includes a -TRE part in charge of removing the trapper instances from the anomalous class. Trapper instances are anomalous instances having a high resemblance with normal instances, often located in dense regions of normal instances. If not removed, these instances would trap the classifier to detect the surrounding normal instances as anomalous (FPs). Thus, trapper removal (-TRE part) contributed to reducing the number of FPs. The idea of trapper removal is supported by our formulated opportunistic approach, where it is sufficient to detect any-behaviour-all-threat. Thus, removing anomalous instances is safe. As described in Chapter 4, each of the base classification methods (RF, XGBoost, SVM) utilised in CD-AMOTRE is applied as a multi-class machine learning method on the aforementioned multi-class data.

In the case of medium data maturity, the insider threat problem is inspected as a one-class data problem, where labelled data for only normal class is available. To meet objective 3, in Chapter 6, we proposed an adaptive one-class ensemble-based anomaly detection framework (E-base-OU), for insider threat detection. In the proposed CD-AMOTRE approach in Chapter 4, class decomposition was introduced to weaken the effect of the normal class. The idea of class decomposition is also applied in the proposed anomaly detection framework in Chapter 6 on the normal class data, however, the target is to detect local anomalous instances (trapper instances) that would not be detected over the whole data. And unlike CD-AMOTRE, the trapper instances in this framework are not removed, however, detected as anomalous behaviours that may refer to malicious insider threat(s). The class decomposition component clusters the normal class data into k clusters, then trains an ensemble of k base anomaly detection methods to construct k models (k ocSVM models or k iForest models). Each model of the k models is trained on a cluster of the k clusters, to identify trapper instances, thus increasing the chance of any-behaviour-all-threat detection. This framework introduced the progressive update component that includes an oversampling procedure to progressively adapt (retrain) the k models with (1) sequential FP chunks in the acquired data and (2) artificial samples of the FP instances in the current FP chunk. This reduced the number of FPs in the upcoming data. The merit of the artificial oversampling procedure is that it is outlier-aware. It intelligently updates the decision boundaries of the k models by avoiding to oversample the outliers from the FPs with respect to the genuine normal class instances. And unlike CD-AMOTRE, the oversampling procedure in this framework is applied on normal instances (FPs), not anomalous instances.

In the case of low data maturity, the insider threat problem was inspected as an online streaming data problem, where there exists no historical labelled data for normal class and anomalous class. To meet objective 4, in Chapter 7, we proposed a data

TABLE 8.2: Summary of results for research contribution approaches.

Chapter	Contribution Approach	Community	TP$_T$/P$_T$	FP (%$FP=\frac{FP}{FP+TN}$ × 100) or FPAlarm
Chapter 4	CD-AMOTRE	com-P	14/16	FP=46 (3.36%)
		com-S	18/21	FP=88 (6.68%)
		com-I	8/12	FP=3 (0.20%)
Chapter 6	E-base-OU	com-P	17/17	FP=50 (18.2%)
		com-S	22/22	FP=49 (18.9%)
		com-I	12/12	FP=43 (14.4%)
Chapter 7	E-RAIDS	com-P	16/17	FPAlarm=0
		com-S	21/22	FPAlarm=0
		com-I	10/12	FPAlarm=0

stream anomaly detection approach, namely E-RAIDS, for real-time insider threat detection. To address streaming data, we suggested a temporary memory – *buffer* – of a predefined capacity (equals to a predefined window size) to accumulate the acquired data instances. A full buffer is processed in an ensemble of p base outlier detection techniques to construct p models (p MCOD models or p AnyOut models). In the proposed anomaly detection framework in Chapter 6, the ensemble approach learns k models, such that each model of the k models is trained on a cluster of the k normal clusters. Unlike the anomaly detection framework, each model of the p models in the proposed E-RAIDS is trained on a feature subspace of the p random feature subspaces. The rationale behind feature subspaces allows to detect local outliers (anomalous instances) that would not be detected over the whole feature space, thus increasing the chance of any-behaviour-all-threat detection. E-RAIDS defined two factors: (1) a survival factor to control the vote of a feature subspace for an alarm at each window iteration; and (2) a vote factor to control the flagging of an alarm at each window iteration. This resulted in reducing the number of false alarms (FPs) flagged. The survival factor was implemented on each feature subspace, so that if an outlier survived for a number of subsequent windows, a subspace votes for an alarm. The vote factor was implemented in the aggregate component of E-RAIDS to combine the ensemble votes and results.

Table 8.2 summarises the best results achieved in each of the mentioned contribution approaches to meet the ultimate aims (Aim 1 and Aim 2) outlined in Section 1.3. The results are shown for each of the communities (com-P, com-S, and com-I) in terms of: (1) the maximum TP$_T$ detected out of P$_T$ malicious insider threats (Aim 1); and (2) the minimum FP (or FPAlarm) attained by the proposed approaches (Aim 2).

Each of the listed contribution approaches in Table 8.2 is proposed based on a certain level of data maturity, nevertheless, we will give a quick comparison of the results obtained. It is evident that E-base-OU in Chapter 6 achieved the maximum TP_T detected out of P_T over all the communities, where it didn't miss any malicious insider threat. However, the percentage of FPs is higher than that of CD-AMOTRE in Chapter 4 over the utilised communities com-P, com-S, and com-I. Although CD-AMOTRE attains the minimum %FP over all communities compared to E-base-OU, CD-AMOTRE misses 1 to 4 threats over the communities in terms of TP_T detected. In Chapter 7, the data stream problem required a different false positive measure to be utilised for evaluation, so FPAlarm over window iterations was defined in Section 7.4.3. E-RAIDS attained a minimum FPAlarm of 0 over the utilised communities. At the same time, E-RAIDS missed only one to two threats over each of the communities.

As a conclusion, we have been able to fulfil all of the aims and objectives outlined in Section 1.3. We summarise the contributions of this thesis as follows:

- Contribution 1: An opportunistic approach – *threat hunting* – to detect insider threats with the aim of any-behaviour-all-threat detection.
- Contribution 2: Defining the insider threat feature space and extracting the feature set to construct community data sets – *data preprocessing*.
- Contribution 3: The *first* class imbalance approach for insider threat detection, namely CD-AMOTRE, for the case of high data maturity.
- Contribution 4: An adaptive one-class ensemble-based anomaly detection framework to detect insider threats, for the case of medium data maturity.
- Contribution 5: A data stream anomaly detection approach, namely E-RAIDS, to detect insider threats in real-time, for the case of low data maturity.

Each of the above mentioned contributions contributed to fulfil the ultimate aims of detecting all malicious insider threats, while reducing the number of false positives (false alarms).

8.2 Future Work

The research work in this thesis opens the door to a number of key research directions. The following extensions can be proposed for our work.

8.2.1 Real-world Insider Threat Data

As discussed in Chapter 4.1, the lack of real world data has been a significant impediment for researchers working on the insider threat problem. The governments and organisations from all domains commonly refuse to share data with researchers or research institutions to protect its users (e.g. employees, business partners, customers, etc.) and assets. The real data is a result of an everyday monitoring and logging process of the activities of users in an organisation, including information related to private user profile, intellectual property, and confidential content [2]. Only in rare cases, such real data may be shared by an organisation under certain data sharing agreements and regulations.

Taking this impediment into consideration, this research was conducted on synthetic insider threat data sets generated by CMU-CERT, following the trend of the current research studies to utilise these data sets.

A recent insider threat data set, namely The Wolf Of Sutd (TWOS) data set [137], has been collected from *real* user interaction with a host machine during the competition organised by Singapore University of Technology and Design (SUTD) in March 2017. The TWOS data set was generated from the activity of 24 users over a period of 5 days through confidentiality agreements with the users. The authors recently announced that a copy of the TWOS data set can be obtained upon meeting several conditions (5 months ago) [138]. Future work includes applying the proposed approaches on the TWOS data set to gauge how they would perform in a more realistic real-life scenario.

The future direction should also target organisations and seek a data sharing agreement in order to get access to real world insider threat data and evaluate the proposed approaches in this thesis. Our interest is ultimately toward financial organisations (institutions) such as banks, insurance companies, and investment companies. As described in Chapter 1.1, there exist three types of insider threats: fraud, sabotage, and IP theft [2]. These types of insider threats put financial institutions at a serious risk.

8.2.2 Broader Impact of Contribution Approaches

The contribution approaches in this thesis are applied to the insider threat problem to detect malicious insider threats. However, these contribution approaches can be applied to other real world problems such as credit card fraud detection, network intrusion detection, fault detection, cancer detection, event detection in sensor networks, and others.

Recall that the class imbalance data problem was addressed in Chapter 4. We proposed a hybrid approach called CD-AMOTRE, which combines class decomposition and a novel oversampling technique, namely AMOTRE, to balance the data distribution of the majority (normal) class and the minority (anomalous) class. As part of future work, CD-AMOTRE can be applied to class imbalance data problems other than the insider threat problem. For example, the cancer problem is a class imbalance data problem where there exists rare malignant (cancer) data among the major distribution of the normal data. The cancer problem is another problem from the medical or health domain, such that the ultimate aim is to detect the rare instances (cancer data), and to reduce the number of false positives. The false diagnosis (FPs) deceives the doctor and would have a huge impact on the patients (e.g. resulting in depression, anxiety, and fear).

Recall also that a one-class anomaly detection problem was addressed in Chapter 6, where only data of normal class is available. We proposed an anomaly detection framework with two components: one-class modelling component and progressive update component. Future work opens a key direction to adopt anomaly detection framework in a real world problem such as the fault detection problem from the control engineering domain. The progressive update component allows the framework to learn from the false faults (FPs), and retrain the models to reduce the future false faults in the system.

Finally in Chapter 7, we addressed online anomaly detection problem over data streams. We proposed E-RAIDS, an ensemble of random subspace anomaly detectors using continuous outlier detection techniques. A real world problem such as event detection in wireless sensor networks would be a future direction for this contribution approach. To ensure reliable detection of critical events in wireless sensor networks, E-RAIDS can continuously adapt the model with new sensor data and reduce the false alarm rate.

8.2.3 Further Experimental Studies

Several directions on utilising alternative methods in the proposed approaches are open. For Chapter 4, future work includes optimising the hyperparameters of the classification methods through metaheuristics methods, such as Genetic Algorithm (GA) for example, which can lead to a higher precision and recall. Also, one can test the proposed concept of *class decomposition* in conjunction with an alternative renowned oversampling technique other than SMOTE, and evaluate its performance compared to CD-SMOTE and CD-AMOTRE. Furthermore, one can adopt the proposed concept of *trapper removal* (-TRE part) in conjunction with a renowned oversampling technique. This would show some empirical evidence of successful hybridisation (e.g. SMOTE-TRE, which is a hybridisation of the -TRE part and the state-of-the-art SMOTE).

For both Chapter 4 and Chapter 6, we can utilise an alternative clustering method for the concept of class decomposition other than k-means clustering. For instance, Nickerson et al. [139] utilised Principal Direction Divisive Partitioning (PDDP) to guide the resampling in imbalanced data sets. PDDP [140] determines the internal structure of a class and has a linear complexity of running time. This work is actually one of the works of renowned Nathalie Japkowicz and her co-authors who focused on *cluster-based sampling* to tackle the within-class imbalance problems [105], [106].

For Chapter 7, future work can include testing the performance of an alternative continuous outlier detection method for data streams, such as STORM [131], Abstract-C [132], COD [126], or other methods. Although these methods were excluded in our experimental studies as argued in Section 7.2.2, incorporating these methods in the proposed approach could show successful performance on insider threat data sets.

In the future work in general, we would suggest to study the sensitivity of the parameters in the different approaches to test the robustness of the models in the presence of uncertainty of these parameters. This would provide us with a view regarding the influence of the parameters and their corresponding tuned values on the models to further optimise the performance of the proposed approaches. Moreover, the 2-fold and 10-fold cross validations may be replaced with leave-one-out cross validation, which may implement (1) leave-one anomalous instance out; or leave-one threat (including the set of anomalous instances associated to it) out.

It is worth also noting that future work may include the study of the time complexity of the proposed contribution approaches, particularly the E-RAIDS approach in Chapter 7 which requires real-time detection of malicious insider threats.

8.2.4 Incorporating 'Big Data'/'Security' Tools

The insider threat problem, especially in the case of data streaming addressed in Chapter 7, leads to a typical 'big data' computational problem. This problem can be addressed by utilising 'big data'/'security' tools, such as Google Cloud Platform [141], which includes cloud storage, high performance infrastructure, data analytics, and machine learning; Splunk [142], which monitors and correlates real-time data in an indexed big data repository to diagnose problems using alerts and dashboards; and E8 security [143], which applies multi-dimensional modelling and machine learning to detect advanced attacks and malicious insider activities.

8.2.5 Addressing the Masquerade Scenario

As discussed in Chapter 4, the proposed CD-AMOTRE showed a lower performance in terms of TP_T measure and FP measure over the community com-I, due to the masquerade scenario (scenario s3). Almost all the malicious insider threats (10 out of 12) in com-I map to scenario s3, where the masquerader steals the identity of the authorised user (victim), accesses the PC machine using the latter's credentials, and sends mass emails to users in the organisation. First, only 50 anomalous behaviours (instances) refer to the 10 malicious insider threats. Therefore, the threats related to scenario s3 are not well represented compared to the other scenarios (more details regarding the other scenarios can be found in Section 4.1). Second, the case of the masquerade scenario in s3 is complex, because the attack is actually executed on the authorised user’s PC machine using their privileges, which deceives the detection system and makes it appear as normal. Hence, the under representation of the anomalous data related to scenario s3 and the complexity of the scenario makes the detection more challenging. This explains the lower performance of CD-AMOTRE over scenario s3 in com-I. We propose to explore how CD-AMOTRE can be extended to address this special masquerade scenario in order to improve its performance in terms of detecting all threats, while reducing FPs.

Bibliography

[1] M. Bishop, K. Nance, and J. Clark, "Introduction to inside the insider threat minitrack", in *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017.

[2] D. M. Cappelli, A. P. Moore, and R. F. Trzeciak, *The CERT guide to insider threats: how to prevent, detect, and respond to information technology crimes (Theft, Sabotage, Fraud)*. Addison-Wesley, 2012.

[3] J. R. Nurse, P. A. Legg, O. Buckley, I. Agrafiotis, G. Wright, M. Whitty, D. Upton, M. Goldsmith, and S. Creese, "A critical reflection on the threat from human insiders–its nature, industry perceptions, and detection approaches", in *International Conference on Human Aspects of Information Security, Privacy, and Trust*, Springer, 2014, pp. 270–281.

[4] C. I. T. Team, "Unintentional insider threats: A foundational study", *Software Engineering Institute Technical Report*, 2013.

[5] C. R. Partners, *2018 insider threat report*, `http://crowdresearchpartners.com/wp-content/uploads/2017/07/Insider-Threat-Report-2018.pdf`, [Online; accessed 27-July-2018].

[6] S. C. Minute, *Sei cyber minute 2017 survey*, `https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=496626`, [Online; accessed 27-July-2018].

[7] M. Collins, "Common sense guide to mitigating insider threats", CARNEGIE-MELLON UNIV PITTSBURGH PA PITTSBURGH United States, Tech. Rep., 2016.

[8] C. M. U. Software Engineering Institute, *2014 u.s. state of cybercrime survey*, `http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=298318`, [Online; accessed 14-February-2018].

[9] ——, *2011 cybersecurity watch survey*, https://www.sei.cmu.edu/news/article.cfm?assetid=52441, [Online; accessed 14-February-2018].

[10] J. Verble, "The nsa and edward snowden: Surveillance in the 21st century", *ACM SIGCAS Computers and Society*, vol. 44, no. 3, pp. 14–20, 2014.

[11] Reuters, *Harold martin (nsa) - theft of ip*, http://www.reuters.com/article/us-usa-cybersecurity-nsa-contractor-idUSKBN15N2N4, [Online; accessed 19-July-2018].

[12] Ascentor, *Anonymous (talktalk) - theft of ip*, http://www.ascentor.co.uk/2015/10/talktalk-cyber-security/, [Online; accessed 19-July-2018].

[13] B. news, *Edward snowden (nsa) - theft of ip*, http://www.bbc.co.uk/news/world-us-canada-23123964, [Online; accessed 19-July-2018].

[14] Computerworld, *Ricky joe mitchell (enervest) - it sabotage*, https://www.computerworld.com/article/2489761/technology-law-regulation/it-pro-gets-4-years-in-prison-for-sabotaging-ex-employer-s-system.html, [Online; accessed 19-July-2018].

[15] A. news, *Bradley manning (united states army) - theft of ip*, https://abcnews.go.com/Politics/bradley-manning-sentenced-35-years-leaking-secrets/story?id=20021288, [Online; accessed 19-July-2018].

[16] B. news, *Jerome kerviel (societe generale) - fraud*, http://www.bbc.co.uk/news/world-europe-27463479, [Online; accessed 19-July-2018].

[17] C. Online, *William sullivan (fis) - fraud*, https://www.csoonline.com/article/2121615/build-ci-sdlc/millions-of-records-stolen-from-fidelity-subsidiary.html, [Online; accessed 19-July-2018].

[18] J. Vijayan, *Computer world cyber crime*, https://www.computerworld.com/article/2534852/cybercrime-hacking/dba-who-stole-consumer-data-gets-57-months-in-prison---4m-bill.html, [Online; accessed 14-February-2018].

[19] B. news, *Donald mackenzie (royal bank of scotland) - fraud*, http://news.bbc.co.uk/1/hi/scotland/5052372.stm, [Online; accessed 19-July-2018].

[20] Computerworld, *Tim lloyd (omega) - it sabotage*, `https://www.computerworld.com/article/2594504/jury-convicts-it-manager-of-crippling-company-s-systems.html`, [Online; accessed 19-July-2018].

[21] B. Böse, B. Avasarala, S. Tirthapura, Y.-Y. Chung, and D. Steiner, "Detecting insider threats using radish: A system for real-time anomaly detection in heterogeneous data streams", *IEEE Systems Journal*, 2017.

[22] P. Parveen, Z. R. Weger, B. Thuraisingham, K. Hamlen, and L. Khan, "Supervised learning for insider threat detection using stream mining", in *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, IEEE, 2011, pp. 1032–1039.

[23] D. Haidar and M. M. Gaber, "Adaptive one-class ensemble-based anomaly detection: An application to insider threats", in *2018 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2018, pp. 1–9.

[24] ——, "Data stream clustering for real-time anomaly detection: An application to insider threats", in *Clustering Methods for Big Data Analytics: Techniques, toolboxes and applications*, O. Nasraoui and C. E. Ben N'cir, Eds. Springer, in press.

[25] ——, "Outlier detection in random subspaces over data streams: An approach for insider threat detection", *Expert Update*, vol. 17, no. 1, pp. 1–16, 2017.

[26] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data", in *Security and Privacy Workshops (SPW), 2013 IEEE*, IEEE, 2013, pp. 98–104.

[27] D. Haidar, *Phd project on github.* `https://github.com/DianaHaidar/PhD-Project`, [Online; accessed 28-July-2018].

[28] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa, "Insight into insiders: A survey of insider threat taxonomies, analysis, modeling, and countermeasures", *arXiv preprint arXiv:1805.01612*, 2018.

[29] I. A. Gheyas and A. E. Abdallah, "Detection and prediction of insider threats to cyber security: A systematic literature review and meta-analysis", *Big Data Analytics*, vol. 1, no. 1, p. 6, 2016.

[30] M. B. Salem, S. Hershkop, and S. J. Stolfo, "A survey of insider attack detection research", *Insider Attack and Cyber Security*, pp. 69–90, 2008.

[31] P. Parveen, *Evolving insider threat detection using stream analytics and big data*. The University of Texas at Dallas, 2013.

[32] I. Rose, N. Felts, A. George, E. Miller, and M. Planck, "Something is better than everything: A distributed approach to audit log anomaly detection", in *Cybersecurity Development (SecDev), 2017 IEEE*, IEEE, 2017, pp. 77–82.

[33] P. A. Legg, O. Buckley, M. Goldsmith, and S. Creese, "Automated insider threat detection system using user and role-based profile assessment", *IEEE Systems Journal*, 2015.

[34] I. Agrafiotis, A. Erola, M. Goldsmith, and S. Creese, "Formalising policies for insider-threat detection: A tripwire grammar", *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 8, no. 1, pp. 26–43, 2017.

[35] A. Ambre and N. Shekokar, "Insider threat detection using log analysis and event correlation", *Procedia Computer Science*, vol. 45, pp. 436–445, 2015.

[36] A. Gamachchi, L. Sun, and S. Boztas, "Graph based framework for malicious insider threat detection", in *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017, pp. 2638,2647.

[37] A. Gamachchi and S. Boztas, "Insider threat detection through attributed graph clustering", in *Trustcom/BigDataSE/ICESS, 2017 IEEE*, IEEE, 2017, pp. 112–119.

[38] P. Moriano, J. Pendleton, S. Rich, and L. J. Camp, "Insider threat event detection in user-system interactions", in *Proceedings of the 2017 International Workshop on Managing Insider Security Threats*, ACM, 2017, pp. 1–12.

[39] W. EBERLE, J. GRAVES, and L. HOLDER, "Insider threat detection using a graph-based approach", *Journal of Applied Security Research*, vol. 6, pp. 32–81, 2011.

[40] L. Huang and M. Stamp, "Masquerade detection using profile hidden markov models", *computers & security*, vol. 30, no. 8, pp. 732–747, 2011.

[41] K. Tang, M.-T. Zhou, and W.-Y. Wang, "Insider cyber threat situational awareness framework using dynamic bayesian networks", in *Computer Science & Education, 2009. ICCSE'09. 4th International Conference on*, IEEE, 2009, pp. 1146–1150.

[42] M. Mayhew, M. Atighetchi, A. Adler, and R. Greenstadt, "Use of machine learning in big data analytics for insider threat detection", in *Military Communications Conference, MILCOM 2015-2015 IEEE*, IEEE, 2015, pp. 915–922.

[43] D. S. Punithavathani, K Sujatha, and J. M. Jain, "Surveillance of anomaly and misuse in critical networks to counter insider threats using computational intelligence", *Cluster Computing*, vol. 18, no. 1, pp. 435–451, 2015.

[44] A. Azaria, A. Richardson, S. Kraus, and V. Subrahmanian, "Behavioral analysis of insider threat: A survey and bootstrapped prediction in imbalanced data", *IEEE Transactions on Computational Social Systems*, vol. 1, no. 2, pp. 135–155, 2014.

[45] S. Sen, "Using instance-weighted naive bayes for adapting concept drift in masquerade detection", *International Journal of Information Security*, vol. 13, no. 6, pp. 583–590, 2014.

[46] E. T. Axelrad, P. J. Sticha, O. Brdiczka, and J. Shen, "A bayesian network model for predicting insider threats", in *Security and Privacy Workshops (SPW), 2013 IEEE*, IEEE, 2013, pp. 82–89.

[47] R. M. Barrios, "A multi-leveled approach to intrusion detection and the insider threat", *Journal of Information Security*, vol. 4, no. 01, p. 54, 2013.

[48] M. Kandias, V. Stavrou, N. Bozovic, and D. Gritzalis, "Proactive insider threat detection through social media: The youtube case", in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, ACM, 2013, pp. 261–266.

[49] O. Brdiczka, J. Liu, B. Price, J. Shen, A. Patil, R. Chow, E. Bart, and N. Ducheneaut, "Proactive insider threat detection through graph learning and psychological context", in *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on*, IEEE, 2012, pp. 142–149.

[50] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest", in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, IEEE, 2008, pp. 413–422.

[51] H. Goldberg, W. Young, M. Reardon, B. Phillips, *et al.*, "Insider threat detection in prodigal", in *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017.

[52] W. T. Young, A. Memory, H. G. Goldberg, and T. E. Senator, "Detecting unknown insider threat scenarios", in *Security and Privacy Workshops (SPW), 2014 IEEE*, IEEE, 2014, pp. 277–288.

[53] W. T. Young, H. G. Goldberg, A. Memory, J. F. Sartain, and T. E. Senator, "Use of domain knowledge to detect insider threats in computer activities", in *Security and Privacy Workshops (SPW), 2013 IEEE*, IEEE, 2013, pp. 60–67.

[54] E Ted, H. G. Goldberg, A. Memory, W. T. Young, B. Rees, R. Pierce, D. Huang, M. Reardon, D. A. Bader, E. Chow, *et al.*, "Detecting insider threats in a real corporate database of computer usage activity", in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2013, pp. 1393–1401.

[55] B. A. Alahmadi, P. A. Legg, and J. R. Nurse, "Using internet activity profiling for insider-threat detection.", in *ICEIS (2)*, 2015, pp. 709–720.

[56] I. Agrafiotis, A. Erola, J. Happa, M. Goldsmith, and S. Creese, "Validating an insider threat detection system: A real scenario perspective", in *Security and Privacy Workshops (SPW), 2016 IEEE*, IEEE, 2016, pp. 286–295.

[57] R. Zhang, X. Chen, J. Shi, F. Xu, and Y. Pu, "Detecting insider threat based on document access behavior analysis", in *Asia-Pacific Web Conference*, Springer, 2014, pp. 376–387.

[58] C. Gates, N. Li, Z. Xu, S. N. Chari, I. Molloy, and Y. Park, "Detecting insider information theft using features from file access logs", in *European Symposium on Research in Computer Security*, Springer, 2014, pp. 383–400.

[59] Y. Chen, S. Nyemba, and B. Malin, "Detecting anomalous insiders in collaborative information systems", *IEEE transactions on dependable and secure computing*, vol. 9, no. 3, pp. 332–344, 2012.

[60] Y. Chen and B. Malin, "Detection of anomalous insiders in collaborative environments via relational analysis of access logs", in *Proceedings of the first ACM conference on Data and application security and privacy*, ACM, 2011, pp. 63–74.

[61] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams", *arXiv preprint arXiv:1710.00811*, 2017.

[62] A. Zargar, A. Nowroozi, and R. Jalili, "Xaba: A zero-knowledge anomaly-based behavioral analysis method to detect insider threats", in *Information Security and Cryptology (ISCISC), 2016 13th International Iranian Society of Cryptology Conference on*, IEEE, 2016, pp. 26–31.

[63] P. Parveen, N. Mcdaniel, Z. Weger, J. Evans, B. Thuraisingham, K. Hamlen, and L. Khan, "Evolving insider threat detection stream mining perspective", *International Journal on Artificial Intelligence Tools*, vol. 22, no. 05, p. 1 360 013, 2013.

[64] M. B. Ahmad, M. Fahad, A. W. Khan, and M. Asif, "A first step towards reducing insider threats in government organizations", *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 16, no. 6, p. 81, 2016.

[65] M. B. Ahmad, A. Akram, M. Asif, *et al.*, "Towards a realistic risk assessment methodology for insider threats of information misuse", in *Frontiers of Information Technology (FIT), 2014 12th International Conference on*, IEEE, 2014, pp. 176–181.

[66] M. B. Ahmad, M. Fahad, A. W. Khan, and M. Asif, "Towards securing medical documents from insider attacks", *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 7, no. 8, 2016.

[67] M. Bin Ahmad, A. Akram, M Asif, and S. Ur-Rehman, "Using genetic algorithm to minimize false alarms in insider threats detection of information misuse in windows environment", *Mathematical Problems in Engineering*, vol. 2014, 2014.

[68] S. Walton, E. Maguire, and M. Chen, “Multiple queries with conditional attributes (qcats) for anomaly detection and visualization”, in *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*, ACM, 2014, pp. 17–24.

[69] P. Legg, N. Moffat, J. R. Nurse, J. Happa, I. Agrafiotis, M. Goldsmith, and S. Creese, “Towards a conceptual model and reasoning structure for insider threat detection”, *Journal of Wireless Mobile Networks ,Ubiquitous Computing ,and Dependable Applications*, vol. 4, no. 4, pp. 20–37, 2013.

[70] C. M. U. CERT Team, *Cmu cert synthetic insider threat data set*, `https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099`, [Online; accessed 12-April-2018].

[71] M. I.o. T. Lincoln Laboratory, *1998 darpa intrusion detection data set*, `https://www.ll.mit.edu/ideval/data/1998data.html`, [Online; accessed 14-February-2018].

[72] A. M. T. Inc., *Amazon mechanical turk*, `https://www.mturk.com/`, [Online; accessed 01-March-2018].

[73] T. Rashid, I. Agrafiotis, and J. R. Nurse, “A new take on detecting insider threats: Exploring the use of hidden markov models”, in *Proceedings of the 2016 International Workshop on Managing Insider Security Threats*, ACM, 2016, pp. 47–56.

[74] A. Keromytis, *Darpa adams data set*, `https://www.darpa.mil/program/anomaly-detection-at-multiple-scales`, [Online; accessed 14-February-2018].

[75] M. Schonlau, *Schonlau masquerading user data set*, `http://www.schonlau.net/intrusion.html`, [Online; accessed 14-February-2018].

[76] P. A. Legg, O. Buckley, M. Goldsmith, and S. Creese, “Caught in the act of an insider attack: Detection and assessment of insider threat”, in *Technologies for Homeland Security (HST), 2015 IEEE International Symposium on*, IEEE, 2015, pp. 1–6.

[77] R. Vilalta, M.-K. Achari, and C. F. Eick, “Class decomposition via clustering: A new framework for low-variance classifiers”, in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, IEEE, 2003, pp. 673–676.

[78] M. Kubat, S. Matwin, *et al.*, “Addressing the curse of imbalanced training sets: One-sided selection”, in *ICML*, Nashville, USA, vol. 97, 1997, pp. 179–186.

[79] G. E. Batista, R. C. Prati, and M. C. Monard, “A study of the behavior of several methods for balancing machine learning training data”, *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[80] K. Yoon and S. Kwek, “An unsupervised learning approach to resolving the data imbalanced issue in supervised learning problems in functional genomics”, in *Hybrid Intelligent Systems, 2005. HIS'05. Fifth International Conference on*, IEEE, 2005, 6–pp.

[81] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smote: Synthetic minority over-sampling technique”, *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[82] H. He, Y. Bai, E. A. Garcia, and S. Li, “Adasyn: Adaptive synthetic sampling approach for imbalanced learning”, in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, IEEE, 2008, pp. 1322–1328.

[83] H. Han, W.-Y. Wang, and B.-H. Mao, “Borderline-smote: A new over-sampling method in imbalanced data sets learning”, *Advances in intelligent computing*, pp. 878–887, 2005.

[84] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, “Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem”, *Advances in knowledge discovery and data mining*, pp. 475–482, 2009.

[85] T. Maciejewski and J. Stefanowski, “Local neighbourhood extension of smote for mining imbalanced data”, in *Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on*, IEEE, 2011, pp. 104–111.

[86] S. Barua, M. M. Islam, X. Yao, and K. Murase, "Mwmote–majority weighted minority oversampling technique for imbalanced data set learning", *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 2, pp. 405–425, 2014.

[87] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems", *J. Mach. Learn. Res*, vol. 15, no. 1, pp. 3133–3181, 2014.

[88] K. Bache and M. Lichman, *Uci machine learning repository*, `https://archive.ics.uci.edu/ml`, [Online; accessed 17-June-2018].

[89] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics", *Information Sciences*, vol. 250, pp. 113–141, 2013.

[90] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, *Keel dataset repository*, `http://www.keel.es/datasets.php`, [Online; accessed 17-June-2018].

[91] ——, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework.", *Journal of Multiple-Valued Logic & Soft Computing*, vol. 17, 2011.

[92] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, "Experimental perspectives on learning from imbalanced data", in *Proceedings of the 24th international conference on Machine learning*, ACM, 2007, pp. 935–942.

[93] L. Breiman, "Random forests", *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[94] H. T. Kam, "Random decision forest", in *Proc. of the 3rd Int'l Conf. on Document Analysis and Recognition, Montreal, Canada, August*, 1995, pp. 14–18.

[95] T. K. Ho, "The random subspace method for constructing decision forests", *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[96] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system", in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, ACM, 2016, pp. 785–794.

[97] J. H. Friedman, "Greedy function approximation: A gradient boosting machine", *Annals of statistics*, pp. 1189–1232, 2001.

[98] ——, "Stochastic gradient boosting", *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[99] J. H. Friedman, B. E. Popescu, *et al.*, "Importance sampled learning ensembles", *Journal of Machine Learning Research*, vol. 94305, 2003.

[100] L. Breiman, "Arcing the edge", Technical Report 486, Statistics Department, University of California at Berkeley, Tech. Rep., 1997.

[101] E. Alpaydin, *Introduction to machine learning*. MIT press, 2014.

[102] C. Cortes and V. Vapnik, "Support-vector networks", *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[103] C. Campbell, "Kernel methods: A survey of current techniques", *Neurocomputing*, vol. 48, no. 1-4, pp. 63–84, 2002.

[104] T. Jo and N. Japkowicz, "Class imbalances versus small disjuncts", *ACM Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 40–49, 2004.

[105] N. Japkowicz, "Concept-learning in the presence of between-class and within-class imbalances", in *Conference of the Canadian Society for Computational Studies of Intelligence*, Springer, 2001, pp. 67–77.

[106] J. Stefanowski, "Dealing with data difficulty factors while learning from imbalanced data", in *Challenges in Computational Statistics and Data Mining*, Springer, 2016, pp. 333–363.

[107] C. G. Varassin, A. Plastino, H. C. da Gama Leitão, and B. Zadrozny, "Undersampling strategy based on clustering to improve the performance of splice site classification in human genes", in *Database and Expert Systems Applications (DEXA), 2013 24th International Workshop on*, IEEE, 2013, pp. 85–89.

[108] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm", *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[109] S. Banitaan, A. B. Nassif, and M. Azzeh, "Class decomposition using k-means and hierarchical clustering", in *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*, IEEE, 2015, pp. 1263–1267.

[110] Š. Brodinová, M. Zaharieva, P. Filzmoser, T. Ortner, and C. Breiteneder, "Clustering of imbalanced high-dimensional media data", *Advances in Data Analysis and Classification*, pp. 1–24, 2017.

[111] J. Wu, H. Xiong, P. Wu, and J. Chen, "Local decomposition for rare class analysis", in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2007, pp. 814–823.

[112] J. R. Quinlan, "C4. 5: Programming for machine learning", *Morgan Kauffmann*, vol. 38, p. 48, 1993.

[113] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers", in *ACM sigmod record*, ACM, vol. 29, 2000, pp. 93–104.

[114] M. Kuhn, "Building predictive models in r using the caret package", *Journal of Statistical Software*, vol. 28, no. 5, pp. 1–26, 2008. [Online]. Available: `https://www.jstatsoft.org/article/view/v028i05`.

[115] F. L. Greitzer and T. A. Ferryman, "Methods and metrics for evaluating analytic insider threat tools", in *Security and Privacy Workshops (SPW), 2013 IEEE*, IEEE, 2013, pp. 90–97.

[116] M. D. Guido and M. W. Brooks, "Insider threat program best practices", in *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, IEEE, 2013, pp. 1831–1839.

[117] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey", *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[118] X. Zhu, "Semi-supervised learning", in *Encyclopedia of machine learning*, Springer, 2011, pp. 892–897.

[119] D. M. Hawkins, *Identification of outliers*. Springer, 1980, vol. 11.

[120] I. Assent, P. Kranen, C. Baldauf, and T. Seidl, "Anyout: Anytime outlier detection on streaming data", in *International Conference on Database Systems for Advanced Applications*, Springer, 2012, pp. 228–242.

[121] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: An efficient data clustering method for very large databases", in *ACM Sigmod Record*, ACM, vol. 25, 1996, pp. 103–114.

[122] C. C. Aggarwal, S. Y. Philip, J. Han, and J. Wang, "-a framework for clustering evolving data streams", in *Proceedings 2003 VLDB Conference*, Elsevier, 2003, pp. 81–92.

[123] M. Hahsler, M. Bolanos, J. Forrest, *et al.*, "Introduction to stream: An extensible framework for data stream clustering research with r", *Journal of Statistical Software*, vol. 76, no. 14, pp. 1–50, 2017.

[124] M. Khalilian and N. Mustapha, "Data stream clustering: Challenges and issues", *arXiv preprint arXiv:1006.5261*, 2010.

[125] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. De Carvalho, and J. Gama, "Data stream clustering: A survey", *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, p. 13, 2013.

[126] M. Kontaki, A. Gounaris, A. N. Papadopoulos, K. Tsichlas, and Y. Manolopoulos, "Continuous monitoring of distance-based outliers over data streams", in *2011 IEEE 27th International Conference on Data Engineering*, IEEE, 2011, pp. 135–146.

[127] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.", in *Kdd*, vol. 96, 1996, pp. 226–231.

[128] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers", *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1641–1650, 2003.

[129] E. M. Knox and R. T. Ng, "Algorithms for mining distancebased outliers in large datasets", in *Proceedings of the International Conference on Very Large Data Bases*, Citeseer, 1998, pp. 392–403.

[130] D. Georgiadis, M. Kontaki, A. Gounaris, A. N. Papadopoulos, K. Tsichlas, and Y. Manolopoulos, "Continuous outlier detection in data streams: An extensible framework and state-of-the-art algorithms", in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, ACM, 2013, pp. 1061–1064.

[131] F. Angiulli and F. Fassetti, "Distance-based outlier queries in data streams: The novel task and algorithms", *Data Mining and Knowledge Discovery*, vol. 20, no. 2, pp. 290–324, 2010.

[132] D. Yang, E. A. Rundensteiner, and M. O. Ward, "Neighbor-based pattern detection for windows over streaming data", in *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, ACM, 2009, pp. 529–540.

[133] T. U.o. W. MOA Team, *Massive online analysis open source framework*, `https://moa.cms.waikato.ac.nz/`, [Online; accessed 14-February-2018].

[134] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "Self-adaptive anytime stream clustering", in *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, IEEE, 2009, pp. 249–258.

[135] A. Guttman, *R-trees: A dynamic index structure for spatial searching*, 2. ACM, 1984, vol. 14.

[136] T. Seidl, I. Assent, P. Kranen, R. Krieger, and J. Herrmann, "Indexing density models for incremental learning and anytime classification on data streams", in *Proceedings of the 12th international conference on extending database technology: advances in database technology*, ACM, 2009, pp. 311–322.

[137] A. Harilal, F. Toffalini, J. Castellanos, J. Guarnizo, I. Homoliak, and M. Ochoa, "Twos: A dataset of malicious insider threat behavior based on a gamified competition", in *Proceedings of the 2017 International Workshop on Managing Insider Security Threats*, ACM, 2017, pp. 45–56.

[138] I. Homoliak, *Twos data set on github.* `https://github.com/ivan-homoliak-sutd/twos`, [Online; accessed 29-July-2018].

[139] A. Nickerson, N. Japkowicz, and E. E. Milios, “Using unsupervised learning to guide resampling in imbalanced data sets.”, in *AISTATS*, 2001.

[140] D. Boley, “Principal direction divisive partitioning”, *Data mining and knowledge discovery*, vol. 2, no. 4, pp. 325–344, 1998.

[141] Google, *Google cloud platform*, `https://cloud.google.com/`, [Online; accessed 28-February-2018].

[142] S. Inc., *Splunk*, `https://www.splunk.com/`, [Online; accessed 28-February-2018].

[143] E. S. Inc., *E8 security*, `http://e8security.com/tag/big-data-platform/`, [Online; accessed 28-February-2018].

Appendices

Adaptive One-Class Ensemble-based Anomaly Detection: An Application to Insider Threats

Diana Haidar and Mohamed Medhat Gaber
School of Computing and Digital Technology Birmingham City University Birmingham, United Kingdom
Email: {diana.haidar,mohamed.gaber}@bcu.ac.uk

Abstract*—The malicious insider threat is getting increased concern by organisations, due to the continuously growing number of insider incidents. The absence of previously logged insider threats shapes the insider threat detection mechanism into a one-class anomaly detection approach. A common shortcoming in the existing data mining approaches to detect insider threats is the high number of False Positives (FP) (i.e. normal behaviour predicted as anomalous). To address this shortcoming, in this paper, we propose an anomaly detection framework with two components: one-class modelling component, and progressive update component. To allow the detection of anomalous instances that have a high resemblance with normal instances, the one-class modelling component applies class decomposition on normal class data to create k clusters, then trains an ensemble of k base anomaly detection algorithms (One-class Support Vector Machine or Isolation Forest), having the data in each cluster used to construct one of the k base models. The progressive update component updates each of the k models with sequentially acquired FP chunks; segments of a predetermined capacity of FPs. It includes an oversampling method to generate artificial samples for FPs per chunk, then retrains each model and adapts the decision boundary, with the aim to reduce the number of future FPs. A variety of experiments is carried out, on synthetic data sets generated at Carnegie Mellon University, to test the effectiveness of the proposed framework and its components. The results show that the proposed framework reports the highest F1 measure and less number of FPs compared to the base algorithms, as well as it attains to detect all the insider threats in the data sets.

I. Introduction

Anomaly detection tackles the rare-class problem by building a model based only on the normal class label, then predicting whether acquired new data is normal or anomalous. This class of anomaly detection algorithms is referred to as semi-supervised anomaly detection in [1]. However, semi-supervised classification refers to those techniques that operate having only a small set of labelled instances along with a typically larger set of unlabelled ones, regardless of the assigned class [2]. Thus, in this paper, we use the term anomaly detection to refer to the one-class machine learning problem.

The ongoing monitoring and logging of the insiders' activities establishes huge useful data sets of information to learn the normal baseline of behaviour. However, the absence of previously logged malicious insider threats, among the logs of normal users' behaviour in an organisation, shapes the insider threat detection mechanism into a one-class data mining approach, namely anomaly detection. The topic of insider threat detection is getting increased concern by organisations, as a result of the significant number of malicious insider threats reported in recent years [3]. These threats are attributed to insiders; current or former employees, contractors, or business partners in an organisation, who have privileged access to the network, system, and data. The risk of the malicious activities carried by insiders is worth a considerable attention more than that of outsiders, due to the horrendous costly corruption that it causes to an organisation. Most of the security mechanisms implemented in an organisation usually tackle the outsider attacks (e.g. anti-viruses, firewalls, intrusion prevention and detection systems), which fortunately reduces the risk of outsiders. However, the privileges given to insiders make the detection of the malicious insider threats more challenging. The users within an organisation are aware of the system and have authorised access to sensitive information, which, if disclosed, will result in costly consequences.

The machine learning approaches proposed for detecting insider threats still have a common shortcoming, which is the great number of false alarms flagged [4], [5], deceiving the administrator(s) about suspicious behaviour of many users. This consumes a valuable time from the administrator's schedule, while investigating the suspected users. On the other hand, it impacts the level of trust between the suspected users and their senior executives in an organisation. A recent Real-time Anomaly Detection In Streaming Heterogenity (RADISH) system, based on k Nearest Neighbours was proposed in [4]. The experimental results showed that 92% of the alarms flagged for malicious behaviour are actually benign. We attribute such high number of False Positives (FPs) to the fact that previously proposed machine learning methods attempt to find all suspicious behaviours (instances) associated with any possible insider threat. However, the focus should be on detecting one or more instance(s) of malicious behaviour(s) per threat. Designing machine learning methods with such a relaxing condition, we argue, has the potential to reduce the frequent false alarming problem.

Taking into consideration this relaxing condition, and to tackle this shortcoming of the high number of FPs, in this paper, we propose an anomaly detection framework that consists of two components: one-class modelling component, and progressive update component. First, the role of the one-class modelling component ramifies to decompose the normal class data into k clusters, and to train an ensemble of k

base anomaly detection algorithms (One-class Support Vector Machine –ocsvm– or Isolation Forest –iForest–). Each base model of the ensemble is trained over one cluster, resulting in an ensemble of k base models. This allows the detection of anomalous *trapper* instances that have a high resemblance with normal instances, which would not be detected by the algorithm if trained over the whole normal class data. Second, the role of the progressive update component is to adapt the decision boundary of each base model with sequentially acquired FP chunks. It includes a selective oversampling method to generate artificial samples for FPs per chunk, with the aim to reduce the number of FPs.

The proposed anomaly detection framework provides the following major contributions:

- a class decomposition method on the normal class data to detect the anomalous *trapper* instances;
- a progressive update method with sequentially acquired FP chunks to address the shortcoming of high number of FPs;
- an outlier-aware artificial oversampling method for FPs to avoid model overfitting by intelligently adapting decision boundaries of base models of the ensemble fed with the synthetically oversampled data; and
- a thorough performance evaluation and a statistical significance test of a variety of experiments utilising ocsvm and iForest, validating the effectiveness of class decomposition, progressive update, and oversampling, compared to that of base ocsvm and base iForest.

The rest of the paper is organised as follows. In Section II, we review the related work on the anomaly detection approaches, including ocsvm and iForest approaches, for insider threat detection. In Section III, we propose an anomaly detection framework, with a detailed description of its components. In Section IV, we describe the utilised data sets [6], and we present the experimental setup and the refined versions of evaluation measures. In Section V, we evaluate a variety of experiments, to assess the effectiveness of the proposed framework and its components. Finally, we conclude our paper with a summary in Section VI.

II. Related Work

The approach to address the insider threat problem depends on whether the organisation historically collected system and network logs of the users' activities. If the data is available, it would either consist of normal instances, or normal instances with insider threat instances based on whether insider attacks previously occurred in the organisation. In this paper, we focus on the cases when data logs for only normal behaviour are available. Based on this, the insider threat problem may be addressed from the perspective of anomaly detection. In the following, we give the related work on the anomaly detection approaches for insider threat detection, including the approaches that utilised ocsvm and iForest.

Zargar et al. [7] introduced a Zero-Knowledge Anomaly-based Behavioural Analysis (XABA) method that learns each user's behaviour from raw logs and network traffic in real-time. Gates et al. [8] used the structure of the file system hierarchy and access similarity measure techniques to build user behaviour profiles and detect anomalous behaviour.

To our knowledge, the ensemble-ocsvm proposed by Parveen et al. [5] is the only approach that utilised ocsvm to classify data into normal versus anomalous for detecting malicious insider threats. It acquires data chunks (e.g. daily logs) of a continuous data stream, where it learns a new model for each chunk, and continuously updates the ensemble with the k models having the minimum prediction error. The results showed the superiority of ocsvm over two-class Support Vector Machine (two-class SVM) in terms of detecting threats. Furthermore, the ensemble-based ocsvm with the updating stream concept achieves better performance than ocsvm with no updating. The authors extended their work in a follow-up paper [9], where the ensemble approach was applied on unsupervised Graph-Based Anomaly Detection (GBAD). The ensemble-ocsvm outperformed ensemble-GBAD in terms of FPs.

A recent framework based on a graph approach and iForest was presented by Gamachchi et al. [10] to isolate suspicious malicious insiders from the workforce. The graph approach extracts graph and subgraph properties based on user activities as well as time dependent features to generate input for iForest. iForest then calculates anomaly scores to separate anomalous behaviour of a user from normal behaviour, without profiling normal behaviour.

A further recent unsupervised ensemble-based anomaly detection system named PRODIGAL was presented in [11]; a result of five years work on the insider threat detection problem [12], [13], [14]. iForest is configured as one of the user-day detectors in PRODIGAL to detect complex insider threat scenarios in real user activities.

The above methods have shown merit in addressing the insider threat detection problem, however, as aforementioned, they do suffer from high false alarms. In this paper, we design an approach aiming at false alarm reduction, adopting a number of proposed methods. Details of the proposed approach are given in the following section.

III. Anomaly Detection Approach for Insider Threat Detection

This section identifies the feature space in the insider threat problem and the categories of the feature set extracted. It then presents the proposed anomaly detection framework for insider threat detection, and provides a detailed description of its components.

A. Insider Threat Feature Space

The first step to tackle the insider threat problem is to identify the feature space. In this paper, we utilised the synthetic data sets generated by Carnegie Mellon University - Community Emergency Response Team (CMU-CERT) [6], where different malicious insider threat scenarios are simulated. The data sets log the behaviour of users as system and

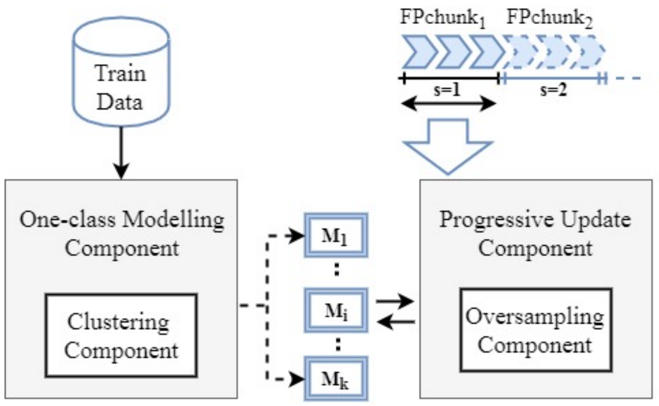


Fig. 1: Conceptual Framework.

network logs (e.g. logons/logoffs, connecting removable media devices, copying files, browsing websites, sending emails, etc.). Based on the literature [4], [15], we extract a feature set from these logs to represent the baseline of users' behaviour including malicious insider threat records. The features used in this work are categorised in four groups: frequency-based (frequency of logon, frequency of connecting device); time-based (logon after work hours, device usage after work hours); Boolean $flag$=$\{0, 1\}$ (non-empty email-bcc, email to a non-employee); attribute-based (browsing a particular URL *job websites, WikiLeaks*); and others (number of email recipients, number of attachments to emails, access to sensitive files based on *file extension*).

Based on the identified feature set, we create community behaviour profiles for users, such that each profile represents the behaviour of users having the same role (e.g. Salesman, IT admin) over session slots. A session slot defines the period of time from *start time* to *end time*, such that the behaviour logs of all users in the community during this period of time are used to extract the identified features. In this work, the session slot is defined per 4 hours, so that the feature set associated to each session slot maps to the community users' behaviour logged during the 4 hours from *start time* to *end time*.

B. Anomaly Detection Framework

In the following, we introduce the proposed anomaly detection framework with its components. Fig. 1 illustrates two phases: one-class modelling component, and progressive update component.

1) One-class Modelling Component: The one-class modelling component acquires training data in the initial modelling phase. It consists of a clustering component that applies k-means clustering algorithm on the normal class data in order to create k clusters. It then trains each cluster on a base algorithm. The result is an ensemble of a base algorithm over k clusters, resulting in k models. In this work, we utilised two highly performing anomaly detection algorithms: ocsvm and iForest, as base algorithms in the proposed framework to detect malicious insider threats. We adopted ocsvm and iForest, because each method has been utilised for insider threat detection, either as a base algorithm for the proposed approaches [5], [9], [10], [11], or as a benchmark against which the performance of a deep learning approach was compared to its performance [16].

a) Clustering Component: Malicious insiders have authorised access to the network, system, and data, and are aware of the system management and security policies. These aspects aid the malicious insider to deceive the detection system, where some anomalous behaviour may have a high resemblance with the normal user's behaviour. This manifests as local anomalous instances located among normal instances. To address this issue, we apply class decomposition [17] on the normal class data. The idea is to decompose the normal class data into clusters and train a detector per cluster, giving more opportunity for the detector to identify local anomalous instances with respect to a cluster which might not be detected over the whole data. We utilise k-means clustering algorithm to identify patterns in the normal class data, given its efficiency.

Let X^t=$\{x_1^t, x_2^t, ..., x_m^t\}$ represent the feature vector at session slot t, where $x_f^t; 1 \preceq f \preceq m$ represents the value of the f^{th} feature. Let y represent the normal class label. Each instance (i.e. feature vector) X^t either belongs or does not belong to normal class y. Let N=X^t $\forall t; X^t \in y$ represent the set of instances which belong to the normal class y. If we apply k-means clustering algorithm on the set N, then N decomposes into k clusters. Let C=$\{C_1, C_2, ..., C_k\}$ represent the set of k clusters.

Fig. 2 represents the normal instances with blue circles, and the anomalous instances with squares. Let the solid-line outer circle represent the decision boundary generated by the base algorithm over the whole normal data to separate the normal instances from the anomalous instances. Consider k=2, so that the normal data instances are grouped into 2 clusters. We construct an ensemble of k base algorithms and train a base algorithm on each cluster of the k clusters. Let the dash-line inner circles represent the decision boundaries generated by each cluster's base algorithm. Fig. 2 reveals *two types of anomalous instances* defined below:

- *Borderline and outlier instances*: represented by red filled squares. Those instances are located at the borderline with respect to the solid-line outer decision boundary, or are far outliers; and
- *Trapper instances*: represented by red empty squares. Those instances are located in a sparse or dense area of normal instances.

The borderline and outlier anomalous instances can be easily detected by one of the aforementioned base algorithms trained over the whole normal data. However, as shown in Fig. 2, the solid-line outer decision boundary would not be able to detect trapper instances. As aforementioned, the trapper instances are located among normal instances, and therefore the base algorithm would declare normal instances (e.g. iForest would not assign a high anomaly score).

To address this issue, we propose to decompose the normal class data into clusters and to train an ensemble of a base

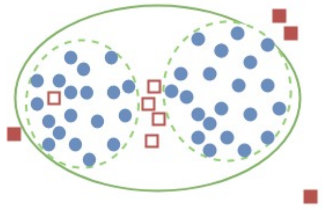


Fig. 2: Clustering normal instances.

algorithm per cluster. So that the trapper instances can be identified by the base algorithm(s) as anomalous with respect to cluster(s). Fig. 2 shows that the dash-line inner decision boundaries, generated by clusters, can detect the trapper instances located in the disjoint area (i.e. sparse area of normal instances).

Nevertheless, there would still exist some trapper instances inside the clusters which may not be detected as positives, due to their existence in an inner dense area of normal instances.

Upon decomposing the normal training data set into k clusters, the role of one-class modelling component ramifies into training an ensemble of a base algorithm on the k clusters to generate k initial models. Let M=$\{M_1, M_2, ..., M_k\}$ represent the set of models generated by the ensemble. Each initial model M_i for $C_i; 1 \leq i \leq k$ is then used to detect the malicious insider threats in the testing data set. The decision d_i^t for a testing instance X^t with respect to M_i comes in Boolean form of {True, False}. In case of ocsvm, an instance X^t either belongs or does not belong to the normal class y. In case of iForest, X^t is identified as an anomalous instance if its anomaly score is greater than a defined anomaly score threshold τ. The parameter τ requires to be tuned as examined later in Section IV.

After that, the ensemble acquires a set of decisions D^t=$\{d_1^t, d_2^t, ..., d_k^t\}$ for a testing instance X^t to vote whether X^t is normal or anomalous. Each decision $d_i^t \in D^t; 1 \leq i \leq k$ is taken by a model M_i for a cluster C_i in the ensemble. The voting mechanism is executed as follows: (1) If $d_i^t \in D^t$ votes for anomalous $\forall i$ (i.e. by all models), then the overall decision D^t declares X^t as anomalous behaviour, and consequently flags an alarm warning of a malicious insider threat; (2) If $\exists d_i^t \in D^t$ votes for normal, then the overall decision D^t declares X^t as normal behaviour.

2) Progressive Update Component: The importance of the insider threat problem requires a continuous monitoring of the implemented detection system by the system's administrator(s). A false alarm is a result of a normal behaviour detected as anomalous by the system. This maps to normal instances that have a high similarity with anomalous instances, thus appear as suspicious events. To address this issue and to minimise the number of false alarms raised, we introduce the progressive update component. The role of this component ramifies to progressively update the detection system with acquired FP chunks. We define an FP chunk as follows:

Definition III.1. FP Chunk An FP chunk is a segment of capacity c that accumulates test instances declared as FPs. Let FPchunk$_s$=$\{$FP^1, FP^2, ..., FP$^c\}$ represent an FP chunk acquired at sequence s, such that D^t for each FP$^t \in$ FPchunk$_s; 1 \leq t \leq c$ is predicted as a positive (i.e. anomalous instance) while the actual class label for FP^t is normal. Thus, each FP^t is declared as FP.

Consider, in Fig. 1, the set of blue arrows represent the testing instances declared as FPs, and each segment of blue arrows represents an FP chunk acquired sequentially. For example, the segment of blue arrows at sequence s=1 represents FPchunk$_1$. We define c as the capacity (size) of FP chunks; each FP chunk FPchunk$_s$ can accumulate c number of FPs. Fig. 1 illustrates FP chunks of capacity c=3; each FP chunk accumulates 3 FPs.

The progressive update method relies on the continuous monitoring by the administrator to investigate whether the flagged alarms are true or false. Along the run of the detection system, when an alarm is flagged, the administrator is required to investigate the suspected user and decide whether it is a malicious insider threat or a false alarm. In the latter case, the FP (false alarm) is accumulated into the FP chunk. This procedure continues until the current FP chunk is full (i.e. capacity c of FP chunk is reached). The FP chunk is then fed to the progressive update component to oversample the FP instances in order to generate artificial samples. The oversampling component is later described in this section. The set of FP instances together with the set of artificial instances are then utilised to update the pre-generated models. Let N represent the pre-generated set of genuine normal instances, and let A_s represent the set of artificial instances generated for FPchunk$_s$. Hence, the pre-generated models are retrained on R=$N\cup$ FPchunk$_s \cup A_s$. Similarly, this process is repeated progressively for each accumulated FP chunk.

The rationale behind the progressive update method is not simply to retrain the models with new instances, however, to enrich the models with recently detected FPs and synthetically generated artificial samples *close (not replicates)* to FPs. So that the decision boundary in each model adapts to the falsely detected behaviour and minimises the chances of FPs in the upcoming testing instances.

The idea of continuous monitoring to investigate flagged alarms to identify FPs has been applied in [16]. The authors defined cumulative recall measure based on a *daily budget*. The term *daily budget* refers to maximum number of alarms an analyst can investigate per day to judge whether it is TP or FP.

a) Oversampling Component: The method of updating pre-generated models with FP instances in the FP chunks may not have a significant influence on the decision boundary. However, the oversampling of FP instances generates artificial instances, and enriches the updated model with recently acquired FP instances as well as normal artificial samples. In this way, the recently acquired normal behaviour of a user or a community will be well represented, and in turn will trigger the base algorithm to adapt the model's decision boundary.

The oversampling component is in charge of generating artificial samples from the FP instances upon the acquirement of each FPchunk$_s$. The task of allocating the number of samples to be generated for each FP instance in the FPchunk$_s$ depends on the degree of outlierness of each FP instance. Thus, the number of samples to be generated varies among FP instances. We utilise a density-based method, namely, Local Outlier Factor (LOF) [18], to calculate the local outlier factor (score) lof_N^t for each FP instance FP^t in acquired FPchunk$_s$ with respect to the k nearest neighbours from only the set of genuine normal class instances N. lof_N^t is tuned for k=$\sqrt{1+card(N)}$ (thumb-rule), where the radicand $1+card(N)$ represents the number of instances utilised to calculate lof_N^t. The motivation to integrate LOF to calculate the anomaly score for FP instances, though iForest can provide it, is that LOF can be used independently of the base algorithm (ocsvm or iForest).

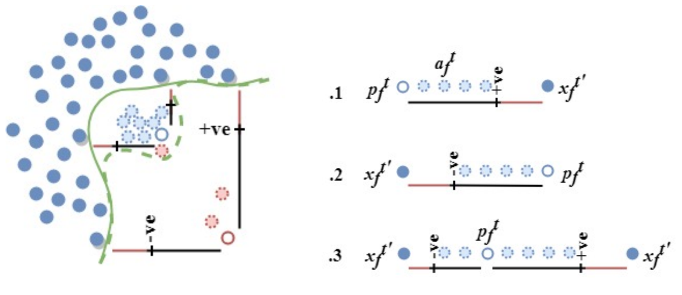
Let $perclof_N^t$ represent the percentile rank for each FP^t compared to the set of normal instances N. In Fig. 3a, we represent the genuine normal instances N by blue filled circles. We define two *types of FP instances*, where each type is oversampled based on its degree of outlierness lof_N^t:

- *Outlier instance*: represented by a solid-line red empty circle. Each FP instance FP^t is considered an outlier instance, if it has a **high** lof_N^t value; located far away from the genuine normal instances N. For example, if $perclof_N^t$=90, this means that the lof_N^t for FP^t is greater than 90% of the normal instances N.
- *Safe instance*: represented by a solid-line blue empty circle. Each FP instance FP^t is considered a safe instance, if it has a **low** lof_N^t value; located at or near the borderline of genuine normal instances N.

In Fig. 3a, the dash-line red circles represent the artificial samples generated for FP outlier instances, while the dash-line blue circles represent the artificial samples for FP safe instances. The idea is that safe instances are given more chance to generate artificial samples around them, while the outlier instances are given less chance. This gives the update component more conservative control on the adaptation of the decision boundary. Oversampling more safe instances than outlier instances safeguards the system from fast movement of the decision boundary due to outliers. Otherwise, more False Negatives (FN) (i.e. anomalous instances predicted as normal) in the upcoming FP chunks.

Let $perc.over$ represent the percentage of artificial samples to be generated, and let $numS$=$(perc.over/100)\times c$ represent the number of artificial samples to be generated. The process of generating artificial samples associated to each FP$^t\in FPchunk_s$ instance is executed feature wise over a number of iterations.

Recall that $x_f^{t'}$ represents the value of the f^{th} feature of X$^{t'}$ at session slot t'. Likewise, let p_f^t represent the value of the f^{th} feature of FP^t at session slot t, given that FP^t=$\{p_1^t, p_2^t, ..., p_m^t\}$. For each feature $f; 1\preceq f\preceq m$, we find the nearest neighbour



(a) Over two features. (b) Over one feature.

Fig. 3: Artificial oversampling of FP instances.

$x_f^{t'}$ of $X^{t'}\in N$ for p_f^t of FP^t. In other words, we search the set of normal instances N at the level of feature f only, and we find the closest feature $x_f^{t'}$ for p_f^t. At the level of feature f, there exists two directions: positive $(+ve)$, and negative $(-ve)$. Thus, p_f^t may have (1) only $+ve$ neighbours from the set N, (2) only $-ve$ neighbours, or (3) both $+ve$ neighbours and $-ve$ neighbours. We define the positive $(+ve)$ nearest neighbour as follows:

Definition III.2. Positive nearest neighbour A $+ve$ nearest neighbour is the closest $x_f^{t'}$ of $X^{t'}\in N$ for p_f^t of FP^t, such that $x_f^{t'}$ is located in the $+ve$ direction to p_f^t.

Similarly, the **negative** $(-ve)$ **nearest neighbour** is defined. Fig. 3b illustrates generating artificial samples of FP instances over one feature (i.e. one dimension). Let the blue filled circle represent $x_f^{t'}$, the blue empty circle represent p_f^t, and the dash-line blue circle represent an artificial feature value a_f^t associated to p_f^t. The process of generating an artificial feature value a_f^t is executed as follows:

- If p_f^t has only a $+ve$ nearest neighbour at the level of feature f (Fig. 3b.1), then a_f^t is calculated in the $+ve$ direction along the segment joining p_f^t and $x_f^{t'}$ according to Eq. 1, such that dir= + 1.
- If p_f^t has only a $-ve$ nearest neighbour at the level of feature f (Fig. 3b.2), then a_f^t is calculated in the $-ve$ direction along the segment joining p_f^t and $x_f^{t'}$ according to Eq. 1, such that dir= − 1.
- If p_f^t has both a $+ve$ nearest neighbour and a $-ve$ nearest neighbour at the level of feature f (Fig. 3b.3), then a_f^t can be calculated in the $+ve$ or $-ve$ direction. A random direction dir is selected at each iteration, and a_f^t is calculated in the selected direction dir according to Eq. 1.

$$a_f^t = p_f^t \; + dir\times rand(0:\lambda\times dist(p_f^t, x_f^{t'})) \quad (1)$$

Note that λ, tuned for λ=.8, denotes a parameter that controls the distance permitted to generate artificial features along the segment joining p_f^t and $x_f^{t'}$. The rationale behind this is to generate the artificial samples a bit closer to the FP instances and not the normal instances, so that the adapted decision boundary is influenced by these samples.

Consequently, an artificial sample A^t=$\{a_1^t, a_2^t, ..., a_f^t\}$ associated to an FP^t instance is generated at each iteration. The

TABLE I: Definition of Experiments.

E-ocsvm	E-iForest	**E**nsemble
ocsvm-U	iForest-U	progressive **U**pdate
E-ocsvm-U	E-iForest-U	**E**nsemble+**U**pdate
ocsvm-OU	iForest-OU	**U**pdate+**O**versampling
E-ocsvm-OU	E-iForest-OU	**E**nsemble+**U**pdate+**O**versampling

steps described are repeated for a number of iterations until $numS$ of artificial samples are generated.

IV. Experiments

The experiments are conducted on the CMU-CERT data sets in Windows Server 2016 on Microsoft Azure (RAM $140GB$, OS $64-bits$, CPU Intel Xeon $E5-2673v3$). First, MATLAB $R2016b$ was used to preprocess the data sets and generate community behaviour profiles per session slots of 4 hours. Second, we implemented the experiments in R environment ($R-3.4.1$).

A. Description of the Dataset

For this paper, we used $r5.2$ data set from the insider threat data sets generated by CMU-CERT. This data set logs the behaviour of 2000 employees over 18 months. Unlike the previously released data sets, the communities in the $r5.2$ data set consist of multiple malicious insider threats which map to 4 different scenarios. Among these employees, we extracted the data logs for employees belonging to the following communities: Production line worker (com-P) with 17 malicious insider threats, Salesman (com-S) with 22, and ITAdmin (com-I) with 12. More information regarding CMU-CERT data sets [19] and simulated scenarios can be found in [6].

B. Experimental Setup

In Table I, we define a variety of experiments performed using the aforementioned anomaly detection algorithms. Each experiment is set up to ocsvm or iForest base algorithms with or without the following: ensemble method, progressive update method, and oversampling method. To evaluate the methods, we performed 10 fold cross-validation on each of the utilised communities.

The experiments are tuned for different values of parameters. ocsvm is evaluated for the $kernel$ values: Linear L, Polynomial P, and Radial R. On the other hand, iForest is evaluated for different values of anomaly score threshold τ=$\{.35, .4, .45\}$, given the number of iTrees nt=20 and the subsample size psi. Note that no anomalous instances were isolated for $\tau > .5$ over the utilised CMU-CERT data sets. This reflects the complex patterns of malicious insider threat behaviour in the data set, and the high resemblance of anomalous behaviour with normal behaviour. Although the authors in [20] pointed out the efficiency of subsampling, psi is tuned for the whole sample size without extracting a subsample of the data set. It is either set up to $card(N)$ over the whole normal training data set, or to $card(C_i)$ over each cluster in the ensemble case. The rationale of using the whole sample size in both cases is related to the progressive update method. It assures that all the FP instances in the progressively acquired FP chunks are used to update the iForest model.

Regarding the ensemble method, we tuned the number of clusters k over a set of arbitrarily chosen small values k=$\{2, 4\}$, as the goal is to detect the anomalous trapper instances. However, the results for only k=2 are reported, due to revealing better performance than k=4. The capacity of FP chunks is tuned over the values c=$\{20, 40, 60, 80, 100\}$ to evaluate the effectiveness of updating the model with *early-acquired-early-updated FPs*. Regarding the oversampling method, the oversampling percentage is tuned for only $perc.over$=200 to test the influence of applying oversampling to the FP chunks on progressive update method.

C. Evaluation Measures

The ultimate aim of the proposed framework is to detect all the malicious insider threats, while minimising the number of FPs. We utilised the following measures to evaluate the experiments. First, we define *default* versions of measures that are evaluated per instance (behaviour): **FP** designates the number of normal instances (behaviours) that are detected as anomalous instances. FP is evaluated with respect to the whole testing set (not per chunk); and TN designates the number of normal instances predicted as normal.

Second, we define *refined* versions of measures that are evaluated per threat: **TP**$_T$ is used instead of TP to evaluate the number of threats detected by the system among all the P$_T$ malicious insider threats. TP$_T$ is incremented if at least one anomalous instance (behaviour associated to the threat) is predicted as anomalous; and FN$_T$ is used instead of FN to evaluate the number of insider threats not detected. Note that the rationale behind introducing refined versions of some measures is related to the ultimate aim of the framework, which is to detect all threats (not necessarily all behaviours), but to minimise FPs (all false alarms).

As a result, the **F1** measure is defined based on the values of the above defined measures. Note F1 is not close to 1 due to the use of refined versions of some measures, but this does not reflect low performance. However, knowing that the maximum TP$_T$ (evaluated per threat) is much less than the minimum FP (evaluated per behaviour), the defined F1 measure closer to 0.5 reveals *significant* performance.

V. Results and Discussion

In this section, we present the results of ocsvm experiments and iForest experiments in terms of the pre-defined evaluation measures. We then show the merit of the proposed framework according to the following objectives:

- Testing the statistical significance of the results using Wilcoxon Signed-Rank Test;
- Comparing ocsvm experiments and iForest experiments;
- Assessing the influence of the proposed components on the framework; and
- Assessing the time efficiency of the progressive update component.

A. Results

Fig. 4 presents the variation of F1 measure as a function of FP chunk capacity c for ocsvm and iForest based experiments over the communities. The results are reported with respect to $kernel$ values and anomaly score threshold τ values for ocsvm based experiments and iForest based experiments respectively. Tables III and IV present the minimum FP and the maximum TP$_T$ attained respectively for ocsvm and iForest based experiments over communities. It reports $kernel$ and FP chunk capacity c associated to the minimum FP and maximum TP$_T$ attained for ocsvm based experiments. On the other hand, it reports anomaly score threshold τ and c associated for iForest based experiments. This allows to identify the superior $kernel$ or τ, and to analyse the influence of FP chunk capacity c on the progressive update method.

1) ocsvm Experiments: Over com-P, ocsvm-U outperforms ocsvm as well as all other ocsvm based experiments in terms of F1 measure, FP, and TP$_T$. ocsvm-U achieves the maximum F1=**0.261**;L while reducing the false positives to the minimum FP=**76**;L compared to FP=106 in ocsvm. Knowing that P$_T$=17, ocsvm-U detects all the malicious insider threats TP$_T$=17;P without missing any threat.

Over com-C, E-ocsvm-U and E-ocsvm-OU report better performance $\forall kernel$, where E-ocsvm-U achieves the maximum F1=**0.318**;P with the minimum FP=**89**;P compared to FP=120 in ocsvm. E-ocsvm-OU follows it with F1=0.300;L and FP=90;P. However, E-ocsvm-U attains the maximum TP$_T$=22;P out of 22, unlike the latter which missed one threat.

Over com-I, E-ocsvm-U reports the best performance in terms of all measures compared to other ocsvm based experiments. It achieves the maximum F1=**0.246**;P, while minimising the number of false positives to FP=**52**;P knowing that FP=149 in ocsvm. Furthermore, it attains TP$_T$=12 out of P$_T$=12 (same TP$_T$ for other experiments).

2) iForest Experiments: Over com-P, E-iForest-U and E-iForest-OU gave a significant boost to F1 measure for τ=.45, where it reaches F1=**0.365**,0.337 respectively. E-iForest-U reduces the number of false positives to the minimum FP=**50**;$.45$, however, it detects TP$_T$=16;$.35$, thus missing one threat. On the other hand, E-iForest-OU reaches a minimum FP=57;$.45$, while detecting all the malicious insider threats TP$_T$=17;$.35$.

Over com-C, E-iForest-U reports the best performance in terms of F1 and FP compared to other iForest based experiments. It achieves the highest F1=**0.422**;$.45$, and the minimum FP=**49**;$.45$. However, it reports TP$_T$=21;$.35$, thus missing the detection of one malicious insider threat. It is worth to note that the iForest based experiments with oversampling method (with or without the ensemble method) attain to detect all the threats for τ=.35

Over com-I, E-iForest-U shows s significant performance in terms of all measures. It achieves the maximum F1=**0.358**;$.45$, while minimising the number false positives to FP=**43**;$.45$. It also attains the maximum TP$_T$=12;$\forall\tau$ (same TP$_T$ for other experiments).

TABLE II: Wilcoxon Signed-Rank Test at .05 significance level.

E-ocsvm	ocsvm-U	E-ocsvm-U	ocsvm-OU	E-ocsvm-OU
0.1073	6.101e-09	8.243e-06	8.587e-07	1.687e-06
E-iForest	iForest-U	E-iForest-U	iForest-OU	E-iForest-OU
0.2719	1.945e-04	4.104e-08	**0.1165**	1.986e-05

B. Statistical Significance: Wilcoxon Signed-Rank Test

To test the significance of the results, we use Wilcoxon Signed-Rank Test which compares each pair of experiments. Each ocsvm based experiment and iForest based experiment is compared with base ocsvm and base iForest respectively. Table II tabulates the *p-value* calculated for each experiment at .05 significance level. With respect to ocsvm, all ocsvm based experiments, except E-ocsvm ($0.1073 > .05$), are significantly different from base ocsvm where *p-value* $< .05$. With respect to base iForest, all iForest based experiments are significantly different from base iForest, except for E-iForest and iForest-OU. The results were predictable, because both E-ocsvm and E-iForest show low performance in terms of the evaluated measures.

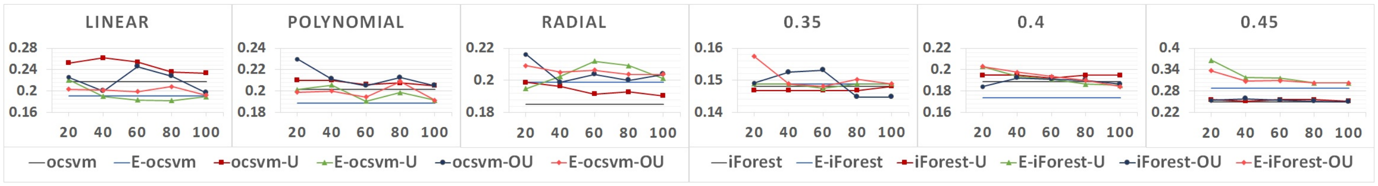
C. Comparing ocsvm Experiments and iForest Experiments

Finally, it is noteworthy that base iForest flagged a lower number of false alarms and reported a higher F1 measure compared to base ocsvm. Moreover, the best performing iForest based experiments over each community achieved the minimal FP compared to ocsvm based experiments. For instance, E-iForest attains the minimum FP=50;$.45, 20$, while ocsvm-U attains FP=76;$L, 40$, compared to FP=106 in base ocsvm over com-P. Over com-S, E-iForest-U reaches the minimum FP=49;$.45, 20$, while E-ocsvm-U and E-ocsvm-OU attain FP=89;$P, 60$ and FP=90 respectively. And lastly, E-iForest-U achieves the minimum FP=43;$.45, 20$, compared to FP=52;$P, 20$ in E-ocsvm-U over com-I. It is apparent that ocsvm works better for $kernel$=$\{L, P\}$, and iForest shows its best performance for τ=.45. Moreover, the less the capacity (size) c of FP chunk, the better the performance of the progressive update method. This emphasises the effectiveness of updating the model with *early-acquired-early-updated FPs* in minimising the number of FPs.

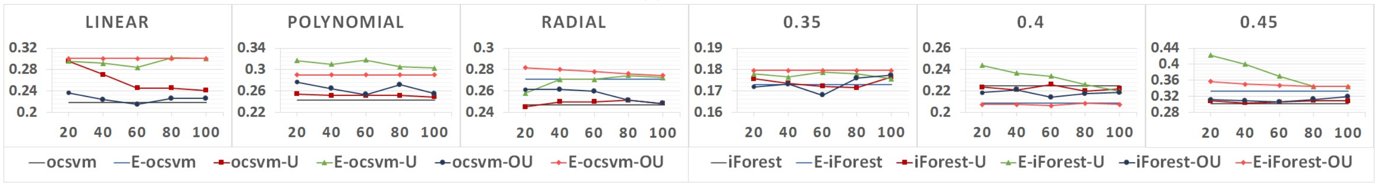
On a side note, the ensemble-ocsvm approach proposed by Parveen et al. [5] reports a percentage of FP rate (%FP=30%). However, our framework reports approximately the half, where %FP varies between (14%-18%) over all the utilised communities.

D. Influence of Proposed Components on the Framework

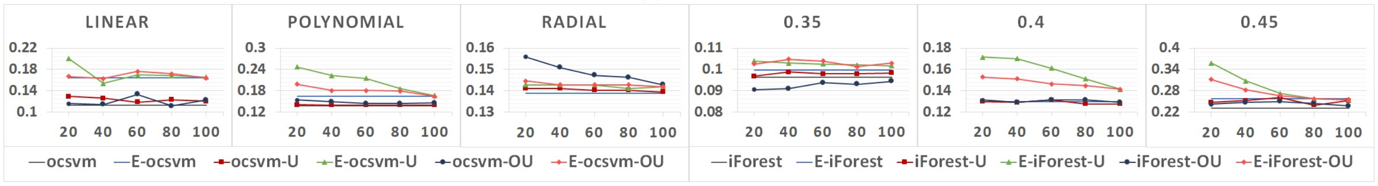
Based on the above extensive experiments carried out to evaluate the influence of the different components on the proposed framework, we sum up what follows. ocsvm-U, E-ocsvm-U, and E-ocsvm-OU outperform other ocsvm based experiments in terms of F1 measure, FP, and TP$_T$ $\forall$ communities. On the other hand, E-iForest-U and E-iForest-OU outperform iForest based experiments $\forall$ communities.



(a) com-P.



(b) com-S.



(c) com-I.

Fig. 4: The variation of F1 measure as a function of FP chunk capacity c for ocsvm and iForest based experiments over communities.

TABLE III: Minimum FP over Communities.

community	ocsvm	E-ocsvm	ocsvm-U	E-ocsvm-U	ocsvm-OU	E-ocsvm-OU
com-P	106 L	97 L	**76** L,40	88 L,20	83 L,60	95 L,80
com-S	120 R	90 P	99 L,20	**89** P,60	109 P,20	**90** P,20
com-I	149 R	100,$\{L, P\}$	146 R,$\{20, 40\}$	**52** P,20	132 P,20	97 P,20
community	**iForest**	**E-iForest**	**iForest-U**	**E-iForest-U**	**iForest-OU**	**E-iForest-OU**
com-P	88 .45	66 .45	85 .45,$\forall c \setminus 100$	**50** .45,20	84 .45,40	**57** .45,20
com-S	85 .45	73 .45	82 .45,$\{20, 100\}$	**49** .45,20	78 .45,100	70 .45,20
com-I	80 .45	69 .45	68 .45,60	**43** .45,20	72 .45,60	53 .45,20

TABLE IV: Maximum TP$_T$ of Detected Insider Threats over Communities.

community	ocsvm	E-ocsvm	ocsvm-U	E-ocsvm-U	ocsvm-OU	E-ocsvm-OU
com-P	**17** P	16 R	**17** P,$\forall c$	16 R,$\forall c \setminus 20$	**17** P,$\forall c$	16 R,$\forall c$
com-S	**22** P	21 R	**22** P,$\forall c$	**22** P,$\{20 - 40\}$	**22** L,$\{20, 60\}$ P, 80	21 R, $\forall$
com-I	**12** $\forall kernel$	**12** R	**12** $\forall kernel$,$\forall c$	**12** R,$\forall c$	**12** $\forall kernel$,$\forall c$	**12** P, 20 R,$\forall c$
community	**iForest**	**E-iForest**	**iForest-U**	**E-iForest-U**	**iForest-OU**	**E-iForest-OU**
com-P	16 .35	16 .35	16 .35,$\forall c$	16 .35,$\forall c$	**17** .35,$\{20 - 60\}$	**17** .35,20
com-S	21 .35	21 .35	21 .35,$\forall c$	21 .35,$\forall c$	**22** .35,$\forall c \setminus 60$	**22** .35,$\forall$
com-I	**12** $\forall \tau$	**12** $\forall \tau$	**12** $\forall \tau$,$\forall c$	**12** $\forall \tau$,$\forall c$	12 $\forall \tau$,$\forall c$	**12** $\forall \tau$,$\forall c$

The importance of the clustering component and the progressive update component as a whole is quite evident, where the number of FPs reached its minimal in experiments which utilised the ensemble method and the progressive update method with FP chunks (e.g., E-ocsvm-U, E-ocsvm-OU, E-iForest-U, E-iForest-OU). We deduce that the effectiveness of the proposed framework relies on the joint collaboration of both components. The use of the clustering component solely, in E-ocsvm and E-iForest, did not improve the performance significantly in terms of the evaluated measures. The Wilcoxon test supported this as revealed in Table II. However, the joint use of progressive update method with ensemble method showed outstanding performance in terms of minimising the number of FPs significantly and achieving higher F1 measure.

Furthermore, we can infer, from Table IV, the support of the oversampling method in detecting all the malicious insider threats over all communities without missing any threat. It is quite remarkable that E-ocsvm-OU and E-iForest-OU show competitive performance to E-ocsvm-U and E-iForest over com-S and com-P respectively. This reveals the potential of the oversampling component.

E. Time Efficiency of Progressive Update Component

Regarding the time complexity, the merit of the proposed progressive update method is its time efficiency. Let ta_s represent the time to accumulate an FPchunk$_s$. We hypothesise that the time to update the ensemble models with the current FPchunk$_s$ is much less than the time to accumulate FPchunk$_{s+1}$ (ta_{s+1}). This ensures that the progressive update of the models is synchronised with the accumulation of sequentially acquired FP chunks.

Recall that, an instance X^t from the acquired test instances is accumulated in the FP chunk, if X^t is declared FP. The FP chunk is progressively accumulated until the capacity c is reached. Hence, the time to accumulate an FP chunk FPchunk$_{s+1}$ actually depends on, (1) n: number of acquired test instances after FPchunk$_s$ until capacity c of FPchunk$_{s+1}$ is reached, and (2) $slot$: period of a session slot for a test instance. Analytically, ta_{s+1}=$n \times slot$.

Consider c=20 and $slot$=4 hours. If n=20, this means that c=20 FPs are accumulated in FPchunk$_{s+1}$ after 20 test instances are acquired. Thus, ALL the n acquired test instances are actually FPs. In this case, ta_{s+1}=20$\times$4=80 hours (approximately 3 days). This case represents the *worst case scenario*, where the next FPchunk$_{s+1}$ is accumulated while 80 hours (3 days) are available to update the models with the current FPchunk$_s$. 80 hours is more than enough to update the models using any state-of-the-art cloud facility. Nevertheless, it would require much less than 80 hours. Hence, the hypothesis is verified.

In the aforementioned worst case scenario, the progressive update is required to run after 80 hours. However, in a typical scenario where an FP chunk is accumulated after $n > c$ number of test instances is acquired, the progressive update will run much less frequently.

VI. Conclusion

The malicious insider threats are getting increased concern by organisations, due to the continuously growing number of insider incidents. A common shortcoming in the proposed detection approaches is the high number of FPs. In this paper, we address this shortcoming based on the availability of only data for normal behaviour, with no previously logged insider incidents.

We propose an adaptive one-class ensemble-based anomaly detection framework with a progressive artificial oversampling method of FPs in class decomposed data. First, a class decomposition method clusters the normal class data, so that an ensemble of k base algorithms (ocsvm or iForest) is trained over the clusters. This allows to detect anomalous *trapper* instances. Second, a progressive update method updates the pre-generated models with progressively acquired FP chunks. This allows the models to benefit from already declared FPs, in order to reduce FPs in upcoming data. Third, an oversampling method is integrated to progressively enrich the models with artificial samples of FPs.

We evaluate the proposed framework in a variety of experiments (with and without: class decomposition/progressive update/oversampling) using ocsvm and iForest. The results show the influence of each method on the performance in terms of higher F1 measure, lower FP, and detecting all malicious insider threats. Moreover, iForest based experiments report a better F1 and a lower FP compared to that of ocsvm.

References

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[2] X. Zhu, "Semi-supervised learning," in *Encyclopedia of machine learning*. Springer, 2011, pp. 892–897.

[3] J. R. Nurse, P. A. Legg, O. Buckley, I. Agrafiotis, G. Wright, M. Whitty, D. Upton, M. Goldsmith, and S. Creese, "A critical reflection on the threat from human insiders–its nature, industry perceptions, and detection approaches," in *International Conference on Human Aspects of Information Security, Privacy, and Trust*. Springer, 2014, pp. 270–281.

[4] B. Böse, B. Avasarala, S. Tirthapura, Y.-Y. Chung, and D. Steiner, "Detecting insider threats using radish: A system for real-time anomaly detection in heterogeneous data streams," *IEEE Systems Journal*, 2017.

[5] P. Parveen, Z. R. Weger, B. Thuraisingham, K. Hamlen, and L. Khan, "Supervised learning for insider threat detection using stream mining," in *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*. IEEE, 2011, pp. 1032–1039.

[6] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *Security and Privacy Workshops (SPW), 2013 IEEE*. IEEE, 2013, pp. 98–104.

[7] A. Zargar, A. Nowroozi, and R. Jalili, "Xaba: A zero-knowledge anomaly-based behavioral analysis method to detect insider threats," in *Information Security and Cryptology (ISCISC), 2016 13th International Iranian Society of Cryptology Conference on*. IEEE, 2016, pp. 26–31.

[8] C. Gates, N. Li, Z. Xu, S. N. Chari, I. Molloy, and Y. Park, "Detecting insider information theft using features from file access logs," in *European Symposium on Research in Computer Security*. Springer, 2014, pp. 383–400.

[9] P. Parveen, N. Mcdaniel, Z. Weger, J. Evans, B. Thuraisingham, K. Hamlen, and L. Khan, "Evolving insider threat detection stream mining perspective," *International Journal on Artificial Intelligence Tools*, vol. 22, no. 05, p. 1360013, 2013.

[10] A. Gamachchi, L. Sun, and S. Boztas, "Graph based framework for malicious insider threat detection," pp. 2638,2647, 2017.

[11] H. Goldberg, W. Young, M. Reardon, B. Phillips *et al.*, "Insider threat detection in prodigal," in *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017.

[12] E. Ted, H. G. Goldberg, A. Memory, W. T. Young, B. Rees, R. Pierce, D. Huang, M. Reardon, D. A. Bader, E. Chow *et al.*, "Detecting insider threats in a real corporate database of computer usage activity," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 1393–1401.

[13] W. T. Young, H. G. Goldberg, A. Memory, J. F. Sartain, and T. E. Senator, "Use of domain knowledge to detect insider threats in computer activities," in *Security and Privacy Workshops (SPW), 2013 IEEE*. IEEE, 2013, pp. 60–67.

[14] W. T. Young, A. Memory, H. G. Goldberg, and T. E. Senator, "Detecting unknown insider threat scenarios," in *Security and Privacy Workshops (SPW), 2014 IEEE*. IEEE, 2014, pp. 277–288.

[15] P. A. Legg, O. Buckley, M. Goldsmith, and S. Creese, "Automated insider threat detection system using user and role-based profile assessment," *IEEE Systems Journal*, 2015.

[16] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," 2017.

[17] R. Vilalta, M.-K. Achari, and C. F. Eick, "Class decomposition via clustering: a new framework for low-variance classifiers," in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE, 2003, pp. 673–676.

[18] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM sigmod record*, vol. 29, no. 2. ACM, 2000, pp. 93–104.

[19] C. M. U. CERT Team, "Cmu cert synthetic insider threat data sets," https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099, [Online; accessed 04-April-2018].

[20] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 413–422.

Data Stream Clustering for Real-time Anomaly Detection: An Application to Insider Threats

Diana Haidar and Mohamed Medhat Gaber

Abstract Insider threat detection is an emergent concern for academia, industries, and governments due to the growing number of insider incidents in recent years. The continuous streaming of unbounded data coming from various sources in an organisation, typically in a high velocity, leads to a typical Big Data computational problem. The malicious insider threat refers to anomalous behaviour(s) (outliers) that deviate from the normal baseline of a data stream. The absence of previously logged activities executed by users shapes the insider threat detection mechanism into an unsupervised anomaly detection approach over a data stream. A common shortcoming in the existing data mining approaches to detect insider threats is the high number of false alarms/positives (FPs). To handle the big data issue and to address the shortcoming, we propose a streaming anomaly detection approach, namely Ensemble of Random subspace Anomaly detectors In Data Streams (E-RAIDS), for insider threat detection. E-RAIDS learns an ensemble of p established outlier detection techniques (Micro-cluster-based Continuous Outlier Detection –*MCOD*– or Anytime Outlier Detection –*AnyOut*–) which employ clustering over continuous data streams. Each model of the p models learns from a random feature subspace to detect *local outliers*, which might not be detected over the whole feature space. E-RAIDS introduces an aggregate component that combines the results from the p feature subspaces, in order to confirm whether to generate an alarm at each window iteration. The merit of E-RAIDS is that it defines a survival factor and a vote factor to address the shortcoming of high number of FPs. Experiments on E-RAIDS-MCOD and E-RAIDS-AnyOut are carried out, on synthetic data sets including malicious insider threat scenarios generated at Carnegie Mellon University, to test the effectiveness of voting feature subspaces, and the capability to detect (more than one)-behaviour-all-threat in real-time. The results show that E-RAIDS-

Diana Haidar
Birmingham City University, Birmingham, UK e-mail: diana.haidar@bcu.ac.uk

Mohamed Medhat Gaber
Birmingham City University, Birmingham, UK e-mail: mohamed.gaber@bcu.ac.uk

MCOD reports the highest F1 measure and less number of false alarm=0 compared to E-RAIDS-AnyOut, as well as it attains to detect approximately all the insider threats in real-time.

1 Introduction

A data stream is a continuous acquisition of data generated from various source(s) in a dynamic environment, typically in a high velocity, leading to accumulation of large volumes of data. This characterisation leads to a typical Big Data computational problem. The dynamic nature of a data stream imposes a change in the data over time. In real-time data streaming, a change refers to an anomalous data that deviates from the normal baseline (e.g. credit card fraud, network intrusion, cancer, etc.). The ultimate aim of such stream mining problems is to detect anomalous data in real-time.

The absence of prior knowledge (no historical database) is often entangled to a real-time stream mining problem. The anomaly detection system is required to employ unsupervised learning to construct an adaptive model that continuously (1) updates with new acquired data, and (2) detects anomalous data in real-time. The system usually acquires data as segments and identifies the *outliers* in the segment as anomalous. An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism [22]. To detect outliers in a stream mining problem, several approaches have been proposed, nevertheless, unsupervised clustering have been successfully applied to identify the patterns in the data and spot outliers [3]. In this work, we select the insider threat problem as a real-world application to detect malicious insider threats (outliers) in real-time. With the absence of labelled data (no previously logged activities executed by users in an organisation), the insider threat problem poses a challenging stream mining problem.

Insider threat detection is an emergent concern for academia, industries, and governments due to the growing number of insider incidents in recent years. An insider is a current or former employee, contractor, or business partner of an organisation who has authorised access to the network, system, or data (e.g. trade secrets, organisation plans, and intellectual property) [34]. Malicious insider threats are attributed to insiders who exploit their privileges with the intention to compromise the confidentiality, integrity, or availability of the system or data. According to the 2011 Cybersecurity Watch Survey [40], 21% of attacks are attributed to insiders in 2011, while 58% are attributed to outsiders. However, 33% of the respondents inspect the insider attacks to be more costly, compared to 51% in 2010. For instance, Harold Martin, a former top-security contractor at the National Security Agency (NSA), was recently convicted for stealing around 500 million pages of national defence information over the course of 20 years [37]. Earlier in 2013, Snowden's attack was reported as the biggest intelligence leakage in the US [42]. Edward Snowden, a for-

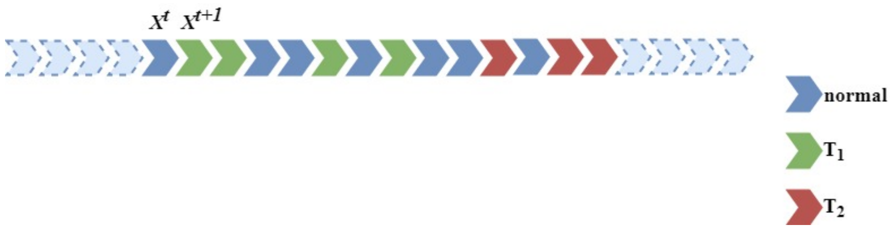


Fig. 1: Continuous data stream of behaviours.

mer contractor to the NSA, disclosed 1.7 million classified documents to the mass media.

The challenge of the insider threat detection problem lies in the variety of malicious insider threats in the data sets. Each malicious insider threat is devised of a complex pattern of anomalous behaviours carried out by a malicious insider, thus making it difficult to detect **all** anomalous behaviours per threat. Analytically, some anomalous instances (behaviours) which exist in a dense area of normal instances have a high similarity to normal instances. These anomalous instances are difficult to detect and may be missed by the detection system.

Based on the challenge of the problem, we formulate this work with the aim to detect **any-behaviour-all-threat**; it is sufficient to detect **any** anomalous behaviour in **all** malicious insider threats. In other words, we can hunt a malicious insider threat by at least detecting one anomalous behaviour among the anomalous behaviours associated to this threat. We call this approach *threat hunting*. The design of the proposed approach with such a relaxing condition contributes in reducing the frequent false alarms.

Fig. 1 illustrates a continuous data stream of behaviours (instances) including normal behaviours and anomalous behaviours. Each arrow denotes a behaviour (instance) X^t executed by a user at a specific period of time. Let the blue arrows represent the normal behaviours. Let the green arrows and the red arrows represent the anomalous behaviours which belong to the malicious insider threats T_1 and T_2 respectively. To detect a malicious insider threat T_1, it is required to detect **any** green behaviour X^t. Hence, it is essential to detect **any** of the anomalous behaviours per malicious insider threat; the aim to detect any-behaviour-all-threat. Note that the proposed approach may detect **more than one** behaviour which belong to a malicious insider threat, nevertheless, the ultimate aim is to detect one anomalous behaviour per threat.

Based on the above argument, the feature space is defined as a set of features which describe the users behaviour. Each feature is extracted from the data set logs to represent a users behaviour related to a particular activity. A feature vector (i.e. instance, behaviour) represent a set of feature values (i.e. actions, commands) evaluated at a period of time. A more detailed description about the feature space is provided in Section 3.

Several machine learning approaches have been suggested to address the insider threat problem. However, these approaches still suffer from a high number of false alarms. A recent real-time anomaly detection system, named RADISH, based on k Nearest Neighbours (k-NN) is proposed by Bose et al. [5]. The experimental results showed that 92% of the alarms flagged for malicious behaviour are actually benign (False Positives –FPs–).

To address the shortcoming of the high number of false alarms, we propose a streaming anomaly detection approach, namely Ensemble of Random subspace Anomaly detectors In Data Streams (E-RAIDS). We presented a preliminary version of E-RAIDS (coined RandSubOut) in [21]. E-RAIDS is built on the top of established outlier detection techniques (Micro-cluster-based Continuous Outlier Detection *–MCOD–* or Anytime Outlier Detection *–AnyOut–*) which employ clustering over continuous data streams. The merit of E-RAIDS is its capability to detect malicious insider threats in real time (*based on the definition of real-time in terms of window iterations as discussed in section* 4).

E-RAIDS learns an ensemble of p outlier detection techniques (either MCOD or AnyOut), such that each model of the p models learns on a random feature subspace. The acquired data is accumulated in a temporary *buffer* of predefined capacity (equals to a fixed window size). So that, at each window iteration, each of the p models in the ensemble is updated with the acquired data, and local outliers are identified in the corresponding feature subspace.

Let p=$n+1$ denote the number of feature subspaces selected randomly, such that n is set to the number of features (dimension) of the feature space F in the community data set. n represents the number of feature pairs (i.e. 2 features per subspace), and 1 represents the whole feature space (i.e. all features). In this way, E-RAIDS is capable to detect *local outliers* in the n feature pairs, as well as *global outliers* in the whole feature space. Hence, E-RAIDS employs the idea of random feature subspaces to detect local outlier(s) (anomalous behaviour(s)) which might not be detected over the whole feature space. These anomalous behaviour(s) might refer to malicious insider threat(s).

E-RAIDS introduces two factors: (1) a survival factor vf_s, which confirms whether a **s**ubspace votes for an alarm, if outlier(s) survive for a vf_s number of window iterations; and (2) a vote factor vf_e, which confirms whether an alarm should be flagged, if a vf_e number of subspaces in the **e**nsemble vote for an alarm. E-RAIDS employs an aggregate component that aggregates the results from the p feature subspaces, in order to decide whether to generate an alarm. The rationale behind this is to reduce the number of false alarms.

The main contributions of this chapter are summarised as follows:

- an ensemble approach on random feature subspaces to detect local outliers (malicious insider threats), which would not be detected over the whole feature space;
- a survival factor that confirms whether outlier(s) on a feature subspace survive for a number of window iterations, to control the vote of a feature subspace;
- an aggregate component with a vote factor to confirm whether to generate an alarm, to address the shortcoming of high number of FPs;

- a thorough performance evaluation of E-RAIDS-MCOD and E-RAIDS-AnyOut, validating the effectiveness of voting feature subspaces, and the capability to detect (more than one)-behaviour-all-threat detection (Hypothesis 2) in real-time (Hypothesis 1).

E-RAIDS extends the preliminary version RandSubOut [21], where it upgrades the selection of the set of outliers (anomalous behaviours) identified at a window iteration over a feature subspace to improve the detection performance of malicious insider threats over the whole ensemble. This is later described in Section 3.3.1. Unlike RandSubOut, we evaluate E-RAIDS on community data sets such that each community is richer with malicious insider threats which map to a variety of scenarios. Moreover, E-RAIDS is evaluated, not only in terms of the number of detected threats and FP Alarms, however, in terms of (1) F1 measure (2) voting feature subspaces, (3) real time anomaly detection, and (4) the detection of (more than One)-behaviour-all-threat.

The rest of this chapter is organised as follows. Sect. 2 reviews the techniques which utilised clustering to detect outliers, and the stream mining approaches proposed for insider threat detection. Sect. 3 presents the proposed streaming anomaly detection approach, namely E-RAIDS, for insider threat detection. Experiments and results are discussed in Sect. 4. Finally, Sect. 5 summarises the chapter and suggests future work.

2 Related Work

The absence of labelled data (i.e. low data maturity) reveals that an organisation has no previously logged activities executed by users (no historical database). We address the absence of prior knowledge using unsupervised streaming anomaly detection built on top of established outlier detection techniques.

Clustering have been successfully applied to identify the patterns of the data for outlier detection [3]. The continuous acquisition of data generated from various sources defines the streaming environment of the insider threat problem. Several clustering methods have been proposed to handle the streaming environment. The state-of-the-art presents two primitive clustering methods: Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [46], and CluStream [1].

BIRCH is an incremental hierarchical clustering method which was first proposed for very large data sets. BIRCH incrementally and dynamically clusters acquired data instances. It maintains a tree of cluster features (information about clusters) which is updated in an iterative fashion [20]. Thereafter, BIRCH was applied to data stream clustering.

BIRCH was the first to introduce the concepts of micro- and macro- clusters [25]. CluStream is a data stream clustering method that employed those two concepts for the clustering process: online micro-clustering, and offline macro-clustering. In the online phase, CluStream scans the acquired data instances and creates micro-clusters in a single pass to handle the big (unbounded) data stream. In the offline phase, CluStream only utilises the micro-clusters and re-clusters into macro-clusters [20].

From BIRCH to CluStream, the concept of data stream clustering is applied, despite the fact that BIRCH includes incremental processing of data instances. *Incremental clustering* processes one data instance at a time and maintain a predefined data structure (i.e. model) that is incrementally updated without the need for model reconstruction [39]. In fact, many incremental methods predate the data stream mining methods. The intrinsic nature of data streams requires methods which implement incremental processing of data instances, in order to handle the resource limitations (i.e. time and memory) [39]. But unlike earlier incremental clustering methods, *data stream clustering* methods require a more efficient time complexity to cope with high data rates [27]. Indeed, the literature stresses the importance of considering the inherent time element in data streams [39]. For instance, a typical data stream clustering method exhibits the temporal aspect of cluster tracking [39]. The dynamic behaviour of cluster over a data stream manifests in the evolving of existing clusters over time; the emergence of new clusters; and the removal of clusters based on a time stamp and/or size. Those cluster updates must be performed on the data structure (i.e. model) very efficiently [27]. Hence, the data stream clustering methods are not only incremental, but are also fast in terms of the inherent temporal aspects.

In this work, data stream clustering is used to underpin the outlier detection techniques for real-time anomaly detection. In the insider threat problem, the temporal aspect is substantial to consider, in order to detect malicious insider threats in real-time (based on the definition of real-time in terms of window iterations as discussed in Section 4). Hence, data stream clustering methods are more adequate to be utilised in this work than typical incremental clustering methods.

In the following we review the techniques which utilised clustering to detect outliers. Thereafter, we shed light on two outlier detection techniques (MCOD and AnyOut) which employ *data stream clustering*. Those two techniques are later utilised in the proposed framework.

We also review the streaming anomaly detection approaches proposed for insider threat detection.

2.1 Clustering for Outlier Detection

It is important to distinguish our objective – to optimise outlier detection – from that of a benchmark of clustering methods developed to optimise clustering (such as DBSCAN [10], BIRCH [46]). The outliers in these methods [10, 46] are referred to as noise, where they are usually tolerated or ignored. However, in our work, we refer to outliers as anomalous (suspicious) behaviour(s) which may correlate to malicious insider threats. Therefore, clustering is utilised to optimise the detection of outliers.

On the other hand, a benchmark of clustering methods to optimise outlier detection (such as LOF [6], CBLOF [23]) are developed. Breunig et al. [6] introduced the concept of Local Outlier Factor (LOF) to determine the degree of outlierness of an instance using density-based clustering. The locality of instance (local density) is estimated by the distance to its k nearest neighbours. Thus, the LOF of an instance located in a *dense* cluster is close to 1, while lower LOF is assigned for other instances. He et al. [23] present a measure, called Cluster-based Local Outlier Factor (CBLOF), to identify the physical significance of an outlier using similarity function. The CBLOF of an instance is determined by (1) the distance to the its nearest cluster (if it belongs to a small cluster), or (2) the distance to its cluster (if it belongs to a large cluster).

However, the aforementioned methods do not tackle outlier detection in *continuous* data streams. MCOD [27] and AnyOut [3] are two established outlier detection techniques, which employ data stream clustering in *continuous* data streams. MCOD and AnyOut are utilised as building block clustering techniques for the proposed E-RAIDS approach. A detailed description and formalisation for these techniques is found in Section 3.2.

2.2 Streaming Anomaly Detection for Insider Threat Detection

Few approaches have utilised streaming anomaly detection to detect insider threats [35, 36, 41, 45] with no prior knowledge. We give a brief description for these approaches in the following.

Among the emerging interest in deep learning, Tuor et al. [41] present a preliminary effort to utilise deep learning in an online unsupervised approach to detect anomalous network activity in real-time. The authors presented two models: a Deep Neural Network model (DNN) which is trained on each acquired instance only once; and a Recurrent Neural Network (RNN) which learns an RNN per user such that the weights are shared among all users, however, the hidden states are trained per user.

Zargar et al. [45] introduce a Zero-Knowledge Anomaly-Based Behavioural Analysis Method, namely XABA, that learns each user's behaviour from raw logs and network traffic in real-time. XABA is implemented on a big-stream platform without predefined or preprocessed activity logs, to handle high rates of network sessions. The authors indicated that XABA reports a low number of FPs, when evaluated on a real traitor scenario.

One of the remarkable approaches is an ensemble of one class SVM (ocSVM), namely Ensemble based Insider Threat (EIT), proposed by Parveen et al. [36]. The authors proposed an ensemble approach, based on static ocSVM learners, to model a continuous data stream of data chunks (i.e. daily logs). The EIT maintains an ensemble of a k predefined number of models M, where the ensemble is continuously updated at each day session upon the learning of a new model M_s. The EIT selects the best $k-1$ models from the k models, having the minimum prediction error over the data chunk C_s, and appends the new model M_s. The results show that ensemble-ocSVM outperforms ocSVM, where it reports a higher accuracy and almost half the number of FP. The authors extended their work in a future paper [35], where the ensemble approach is applied to unsupervised Graph-Based Anomaly Detection

(GBAD). The results show that ensemble-ocSVM outperforms the ensemble-GBAD in terms of FP.

The above approaches have shown merit in addressing the insider threat detection problem, however, as aforementioned, they do suffer from high false alarms. In this book chapter, we utilise data stream clustering to detect outliers (malicious insider threats), while reducing the number of false alarms.

3 Anomaly Detection in Data Streams for Insider Threat Detection

This section identifies the feature space in the insider threat problem and the categories of the feature set extracted. It then describes and formalises established continuous distance-based outlier detection techniques (MCOD and AnyOut). It presents the proposed *E-RAIDS* for insider threat detection with a detailed description of the feature subspace anomaly detection and the aggregate component (ensemble voting).

3.1 Insider Threat Feature Space

In this book chapter, we utilise the synthetic data sets, including a variety of malicious insider threat scenarios, generated by Carnegie Mellon University (CMU-CERT) [15]. The CMU-CERT data sets comprise system and network logs for the activities carried out by users in an organisation over 18 months (e.g. logons, connecting removable devices, copying files, browsing websites, sending emails, etc.). We extract a feature set from these logs in the CMU-CERT data sets according to the literature [5, 29, 30]. This feature set allows us to assess the behaviour of users, and compare it to the previous behaviour of the users or their community of users. We extract each feature considering the evidence it would reveal about an undergoing anomalous behaviour. For example, consider the feature *logon after hours*; this feature if its values is greater than 1, it reveals an evidence of an unusual activity undergoing after the official working hours. Thus, it contributes in the overall decision of the system whether a malicious alarm should be flagged or not.

In the following, we give a more detailed description of the feature set used in this work. We categorise the features into four groups with some examples of each.

- Frequency-based features: assess the frequency of an activity carried out by a community of users over each session slot. (*integer* e.g. *frequency of logon, frequency of connecting devices*);
- Time-based features: assess an activity carried out within the non-working hours period of time. (*integer* e.g. *logon after work hours, device usage after work hours*);

- Boolean features: assess the presence/absence of an activity-related information. ($flag$=$\{0,1\}$ e.g. *non-empty email-bcc, a non-employee email recipient, sensitive file extension*);
- Attribute-based features: are more specialised features, which assess an activity with respect to a particular attribute (feature) value. ($integer$ e.g. *browsing a particular URL (job websites, WikiLeaks)*); and
- Other features: assess the count of other activity-related information. ($integer$ e.g. *number of email recipients, number of attachments to emails*).

This feature set defines the feature space of the insider threat problem, and is used to construct community behaviour profiles for users having the same role (e.g. Salesman, IT admin). A community behaviour profile represents instances (i.e. vectors of feature values) evaluated over session slots, where a session slot represents a period of time from *start time* to *end time*. Each vector of feature values is extracted from the behaviour logs of the community users during a session slot.

In this work, we define the session slot per four hours to find local anomalous behaviour within a day which would not be detected per day. The rationale behind choosing the session slot *per four hours* is that this period of time is long enough to extract an instance (i.e. vector of feature values) which provides an adequate evidence of anomalous behaviour. Thus, it supports the system to capture the anomalous behaviours in feature space. If the session slot is chosen per minutes, for example, the extracted instances would lack the adequate evidence of the occurrence of anomalous behaviour. On the other hand, if the session slot is chosen per days/weeks, for example, the period of time will be too long to capture the anomalous behaviour blurred among the normal behaviour in the extracted vector of feature values.

After constructing the community behaviour profiles, we normalise each vector of feature values (over a session slot) to the range $[0,1]$, and associate it with a class label $\{Normal, Anomalous\}$.

3.2 Background on Distance-based Outlier Detection Techniques

The state-of-the-art presents one of the most widely employed techniques for anomaly detection, which are *distance-based* outlier detection techniques. According to the definition [26], an instance X^t is an outlier, if there exists less than k number of neighbours located at a distance at most r form X^t.

In this work, we are interested in *continuous outlier detection* over a data stream, where recent instances arrive and previous instances expire. The demo paper [14] gives a comparison of four continuous distance-based outlier detection techniques: STream OutlieRMiner (STORM) [2]; Abstract-C [44]; Continuous Outlier Detection (COD) [27]; and Micro-cluster-based Continuous Outlier Detection (MCOD) [27]. COD and MCOD have $O(n)$ space requirements, and they have a faster running time than the exact algorithms of both [2] and [44]. Note that since all algorithms are exact, they output the same outliers [14]. According to [14], COD and MCOD demonstrate a more efficient performance compared to STORM and Abstract-C in terms of lower space and time requirements. Hence, the latter are excluded, but not COD and MCOD. Furthermore, based on the experimental evaluation in [27], MCOD outperforms COD over benchmark tested data sets. Hence, MCOD is selected to be utilised as a base learner in the proposed E-RAIDS approach.

The state-of-the-art presents a further continuous distance-based outlier detection technique, called Anytime Outlier Detection (AnyOut) [3]. However, AnyOut has not been compared to the four techniques in the demo paper. We select MCOD and AnyOut as base learners in E-RAIDS to compare their performance. It is worth to note that the aforementioned distance-based outlier detection techniques are implemented by the authors of [14] in the open-source tool for Massive Online Analysis (MOA) [32].

In the following, a brief description of MCOD and AnyOut techniques is provided.

3.2.1 Micro-cluster-based Continuous Outlier Detection

Micro-cluster-based Continuous Outlier Detection (MCOD), an extension to Continuous Outlier Detection (COD), stems from the adoption of an event-based technique. The distinctive characteristic of MCOD is that it introduces the concept of evolving micro-clusters to mitigate the need to evaluate the range query for each acquired instance X^t with respect to all the preceding active instances. Instead, it evaluates the range queries with respect to the (fewer) centers of the micro-clusters. The micro-clusters are defined as the regions that contain inliers exclusively (with no overlapping). The micro-clustering is fully performed online [27].

Given that, the center mc_i of a micro-cluster MC_i may or may not be an actual instance X^t; the radius of MC_i is set to $\frac{r}{2}$, such that r is the distance parameter for outlier detection; and the minimum capacity (size) of MC_i is $k+1$. Below we give a brief formalisation of MCOD.

Let k represent the number of neighbours parameter. Let PD represent the set of instances that doesn't belong to any micro-cluster.

The micro-clusters (i.e. centers of micro-clusters) in MCOD are determined as described in [1]. In the initialisation step, seeds (with a random initial value) are sampled with a probability proportional to the number of instances in a given micro-cluster. The corresponding seed represents the centroid of that micro-cluster. In later iterations, the center mc_i is the weighted centroid of the micro-cluster MC_i.

- Step 1: For each acquired instance X^t, MCOD finds (1) the nearest micro-cluster MC_i, whose center mc_i is the nearest to it, and (2) the set of micro-cluster(s) R, whose centers are within a distance $\frac{3\times r}{2}$ from their centers.
- Step 2: If $dist(X^t, mc_i) \leq \frac{r}{2}$; such that mc_i is the center of the nearest cluster MC_i, X^t is assigned to the corresponding micro-cluster.
- Step 3: Otherwise, a range query q for X^t is evaluated with respect to the instances in (1) the set PD and (2) the micro-clusters of the set R.

- Step 4: Consider n: the number of neighbours $N^{t'} \in P$ to X^t, such that $dist(X^t, N^{t'}) \geq \frac{r}{2}$. If $n > \theta k; \theta \geq 1$, then a new micro-cluster with center X^t is created and the n neighbours are assigned to this micro-cluster.
- Step 5: A micro-cluster whose size decreases below $k+1$ is deleted and a range query similar to that described for X^t is performed for each of its former instances.
- Step 6: An instance X^t is flagged as an outlier, if there exists less than k instances in either PD or R.

3.2.2 Anytime Outlier Detection

Anytime Outlier Detection (AnyOut) is a cluster-based technique that utilises the structure of ClusTree [28], an extension to R-tree [19, 38], to compute an outlier score. The tree structure of AnyOut suggests a hierarchy of clusters, such that the clusters at the upper level subsume the fine grained information at the lower level. ClusTree is traversed in top-down manner to compute the outlier score of an acquired instance X^t until it is interrupted by the subsequent (next) instance X^t. Thus, the descent down the tree improves the certainty of the outlier score, nevertheless, the later the arrival of the subsequent instance the higher the certainty [3].

Given that, a cluster is represented by a Cluster Feature tuple $CF = (n, LS, SS)$, such that n is the number of instances in the cluster, LS and SS are respectively the linear sum and the squared sum of these instances. The compact structure of the tree using CF tuples reduces space requirements. From BIRCH [46] to CluStream [1], cluster features and variations have been successfully used for online summarisation of data, with a possible subsequent offline process for global clustering.

Let e_s represent a cluster node entry in ClusTree. Given defined two scores to compute the degree of outlierness of an instance X^t: (1) a mean outlier score $s_m(X^t)$=$dist(X^t, \mu(e_s))$, such that $\mu(e_s)$ is the mean of the cluster node entry $e_s \in$ ClusTree where X^t is interrupted; and (2) a density outlier score is $s_d(X^t)$=$1 - g(X^t, e_s)$, such that $g(x_i, e_s)$ is the Gaussian probability density of X^t for $\mu(e_s)$ as defined in [3]. Below we give a brief formalisation of AnyOut.

In the case of a constant data stream:

- Step 1: Initialisation: Each X^t in the data stream is assigned with an actual confidence value $conf(X^t)$=$e^{s(X^t)}$.
- Step 2: Distribute the computation time for each X^t based on the scattered confidences.
- Step 3: For each acquired instance X^t, AnyOut traverses the tree in a top-down manner until the arrival of the instance X^{t+1} in the data stream.
- Step 4: At the moment of interruption, X^t is inserted to the cluster node entry $e \in$ ClusTree, where X^t arrives (pauses).
- Step 5: The outlier score of X^t, according to the specified parameter $s_m(X^t)$ or $s_d(X^t)$, is computed with respect cluster node entry e_s.

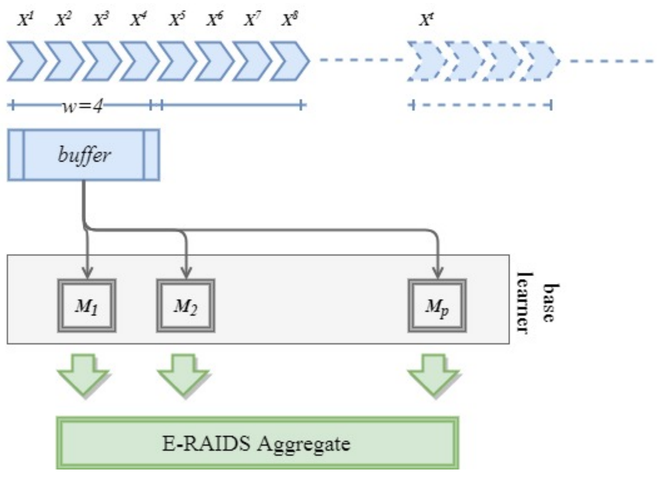


Fig. 2: E-RAIDS framework.

- Step 6: The expected confidence value for the outlier score of X^t is updated based on the computation time. *The confidence value (certainty) increases as the computation time increases*.

3.3 E-RAIDS Approach

The established continuous distance-based outlier detection techniques (MCOD and AnyOut), described and formalised in section 3.2, are utilised as building block data stream clustering techniques for the proposed E-RAIDS approach.

In this work, we propose a streaming anomaly detection approach, namely Ensemble of Random subspace Anomaly detectors In Data Streams (E-RAIDS), for insider threat detection. In other words, E-RAIDS is an ensemble of an established distance-based outlier detection technique (MCOD or AnyOut), such that each model of the ensemble learns on a random feature subspace. The idea of E-RAIDS is to employ an outlier detection technique on a feature subspace, to detect *local outliers* which might not be detected over the whole feature space. These local outliers may refer to anomalous behaviours (instances) attributed to a malicious insider threat. Hence, the ultimate aim of the E-RAIDS approach is to detect any-behaviour-all-threat (*threat hunting* as defined in Section 1); a process that leads to a reduction of the number of false alarms.

Fig. 2 presents the E-RAIDS framework. The set of blue arrows represent a data stream, where each arrow represents an instance (feature vector) X^t acquired at a session slot t. Consider the formalisations below:

- window: a segment of a fixed size w that slides along the instances in a data stream with respect to time (i.e session slots);
- *buffer*: a temporary short memory of allocated capacity (equals to w). It temporarily accumulates the instances in a data stream. The *buffer* starts to accumulate the instances in a data stream at the start of the window and stops once the *buffer* is full after w number of instances. The full *buffer* is then input to the base learner component to be processed; and
- window iteration $wIter$: an iteration of the window slide. It starts at the already processed instances (in the previous *buffer*) $procInst$ and ends at $procInst + w$. For instance, $wIter$=0 starts at $procInst$=0 and ends at w. $wIter$=1 starts at $procInst$=w and ends at $2 \times w$. The window iteration $wIter$ allows the synchronisation between the window slide and the *buffer* accumulation.

Based on the aforementioned formalisations, at each window iteration $wIter$, the *buffer* accumulates w number of instances. Once the *buffer* is full, the instances in the *buffer* are input to the base learner component. A base learner component refers to the distance-based outlier detection technique to be utilised (MCOD or AnyOut). It employs a p number of base models to learn on randomly selected p feature subspaces. A feature subspace $FS_i \subseteq F$ is defined as a subset of features selected from the whole feature space f, where $F = \{f_1, f_2, \ldots, f_n\}$. The rationale behind the idea of random feature subspaces is to detect local outlier(s) (anomalous behaviour(s)) which might not be detected over the whole feature space.

Let p=$n+1$ represent the number of feature subspaces selected randomly, such that n is set to the number of features (dimension) of the feature space F in the community data set. n represents the number of feature pairs (i.e. 2 features per subspace), and 1 represents the whole feature space (i.e. all features). The p feature subspaces are utilised to build the ensemble of p models $\{M_1, M_2, \ldots, M_n, M_p\}$, such that $\{M_1, M_2, \ldots, M_n\}$ learn on feature pairs, and M_p learns on the whole feature space. In this way, E-RAIDS is capable to detect *local outliers* in the n feature pairs, as well as *global outliers* in the whole feature space.

3.3.1 Feature Subspace Anomaly Detection

For each model M_i on a feature subspace FS_i, we define the following data repositories and a survival factor:

- *outSet*: a temporary set of the outliers detected by a base learner (MCOD or AnyOut) at $wIter$;
- *outTempList*: a list that stores the triples (1) an outlier $out \in outSet$, (2) the $wIter$ where it was first detected, and (3) a temporal count $tempC$ which counts the number of subsequent windows out was detected at. It has the form $\langle out, wIter, tempC \rangle$; and
- *subVoteList*: a list that stores the triples (1) a $wIter$, (2) a $subVote$ parameter set to 1 if the feature FS_i votes for an alarm to be generated at $wIter$ and 0 otherwise, and (3) an outlier set $subOutSet$. It has the form $\langle wIter, subVote, subOutSet \rangle$.

- vf_s: a survival factor that confirms whether a feature **s**ubspace FS_i votes for an alarm to be generated at $wIter$. In other words, if an outlier *out* survived (has been detected) for a vf_s number of subsequent windows, then *out* is defined as a *persistent* outlier. A persistent outlier boosts the chance of an alarm to be generated at $wIter$.

Over each feature subspace FS_i, the base learner (MCOD or AnyOut) processes the *buffer* at the current $wIter$ to update the model M_i. M_i identifies the outlier set $outSet$, which includes (1) the outliers from the *buffer* at the current $wIter$; and (2) the outliers from the previously learned instances before the model being updated. The type (2) outlier refers to an instance that was identified as an inlier, however, turned into an outlier in the current $wIter$.

For each outlier $out \in outSet$, if *out* does not exist in $outTempList$, a new triple $\langle out, wIter, 1\rangle$ is appended to $outTempList$. In this case, $tempC$ is assigned to 1 to declare that it is the first time an outlier *out* detected. If *out* exists in the $outTempList$, $tempC$ is incremented by 1 in the triple for out.

For each outlier $out \in outTempList$, E-RAIDS checks (1) if $tempC$=vf_s, then *out* survived for a vf_s number of subsequent windows. We call it *persistent outlier*. Thus, a persistent outlier confirms that the FS_i votes for an alarm at $wIter$. Thus, $subVote$ is set to 1 in the triple for $wIter$ in $subVoteList$. (2) If $wIter - tempC$=vf_s, then *out* has turned into an inlier. We call it *expired outlier*. Thus, the triple for the expired outlier *out* is removed from the $outTempList$. (3) If $wIter - tempC < vf_s$, then *out* is neither a persistent outlier nor an expired outlier. We call it *potential outlier*.

Eventually, *outTempList* consists of persistent outliers and potential outliers (expired outliers has been removed). E-RAIDS appends persistent outliers and potential outliers in *outTempList* to the *subOutSet* in the triple for $wIter$ in $subVoteList$.

In the preliminary version RandSubOut [21], only the persistent outliers in $outTempList$ are appended to the *subOutSet* in the triple for $wIter$ in $subVoteList$. The *subOutSet* at $wIter$ represents the set of anomalous behaviours detected over a feature subspace FS_i. As later described in Section 3.3.2, if the ensemble votes to generate an alarm, then the *subOutSet* for each feature subspace is utilised to evaluate whether all malicious insider threats are detected (i.e. the aim of any-behaviour-all-threat). The experiments carried out on both versions (E-RAIDS and RandSubOut) showed that E-RAIDS outperforms the latter in terms of detecting more malicious insider threats. Analytically, unlike RandSubOut, E-RAIDS considers further the potential outliers in the *subOutSet* to check for detected malicious insider threats. Thus, the use of potential outliers significantly boosts the detection performance of the proposed approach.

Finally, the *buffer* is emptied to be prepared for the subsequent (next) window of upcoming instances ($wIter + 1$). A step-by-step pseudo code for the E-RAIDS base learner procedure is provided in Algorithm 1.

Algorithm 1 Feature Subspace

```
1: $wIter \leftarrow 0$
2: foreach $X^t \in$ stream do
3:     accumulate $X^t$ to $buffer$
4:     if $buffer$ is full then
5:         $outSet \leftarrow$ get outliers detected by base
6:         foreach $out \in outSet$ do
7:             if $outinoutTempList$ then
8:                 a new triple $\langle out, wIter, 1\rangle$ is appended to $outTempList$
9:             else
10:                increment $tempC$ by 1 for $out$ in $outTempList$
11:            end if
12:        end for
13:        foreach $out \in outTempList$ do
14:            if $tempC = vf_s$ then
15:                set $subVote$ to 1 in $subVoteList$ for the current $wIter$
16:                remove out from $outTempList$
17:                $subOutSet \leftarrow$ persistent $out \cup$ potential outliers
18:                append $subOutSet$ to $subVoteList$ at $wIter$
19:            end if
20:            if $wIter - tempC = vf_s$ then
21:                remove $out$ from $outTempList$
22:            end if
23:        end for
24:        increment $wIter$
25:    end if
26:    empty $buffer$
27: end for
28: return $subVoteList$
```

Algorithm 2 Ensemble of Random Feature Subspaces

```
1: foreach $wIter$ do
2:     foreach $FS_i \in FS$ do
3:         $subVote \leftarrow$ get $subVote$ for $wIter$ from $subVoteList$ for $FS_i$
4:         $eVote \leftarrow eVote + subVote$ for $wIter$ in $eVoteList$
5:         $subOutSet \leftarrow$ get $subOutSet$ for $wIter$ from $subVoteList$ for $FS_i$
6:         append $subOutSet$ to $eOutSet$ for $wIter$ in $eVoteList$
7:     end for
8:     if $eVote = vf_e$ then
9:         flag an alarm
10:    end if
11: end for
```

3.3.2 Ensemble of Random Feature Subspaces Voting

For the ensemble of p models $\{M_1, M_2, \ldots, M_n, M_p\}$ on feature subspaces $\{FS_1, FS_2, \ldots, FS_n, FS_p\}$ respectively, we define the following data repository and a vote factor:

- *eVoteList*: a list that stores the triples (1) a $wIter$, (2) an $eVote$ parameter that counts the number of feature subspaces that vote for an alarm, and (3) an outlier set *eOutSet* that appends the *subOutSet* from each FS_i votes for an alarm. It has the form $\langle wIter, subVote, subOutSet \rangle$.
- vf_e: a vote factor that confirms whether an alarm to be generated at $wIter$ by the whole **e**nsemble. In other words, if a vf_e number of feature subspaces vote for an alarm, then an alarm is generated at $wIter$.

As illustrated in Fig. 2, the E-RAIDS aggregate component aggregates the results from the p feature subspaces, in order to confirm whether to generate an alarm or not at each window iteration $wIter$. For each feature subspace FS_i, if $subVote$=1 for $wIter$, E-RAIDS adds $subVote$ to $eVote$ for $wIter$ in *eVoteList*. Furthermore, E-RAIDS gets *subOutSet* for $wIter$ from *subVoteList*, and appends to *eOutSet* for $wIter$ in *eVoteList*.

After getting the votes from all the feature subspaces in the ensemble, E-RAIDS checks if $eVote$=vf_e. If the condition is satisfied, then an alarm of a malicious insider threat is generated at $wIter$. The voting mechanism is technically controlled by the vote factor vf_e, such that if a vf_e number of feature subspaces vote for anomalous behaviour(s) at a window iteration $wIter$, then an alarm is generated. This accordingly handles the case of a conflict, where $\frac{p}{2}$ (50%) of the feature subspaces in the ensemble vote for anomalous behaviour(s) and the other $\frac{p}{2}$ vote for normal behaviour(s). The ensemble technically checks if $\frac{p}{2} = vf_e$, then an alarm is generated.

A step-by-step pseudo code for the E-RAIDS aggregate procedure is provided in Algorithm 2.

4 Experiments

We evaluated the effectiveness of the proposed approach on the CMU-CERT data sets using Windows Server 2016 on Microsoft Azure (RAM 140GB, OS $64-bits$, CPU Intel Xeon $E5-2673v3$) for data pre-processing and Microsoft Windows 7 Enterprise (RAM 12GB, OS $64-bits$, CPU Intel Core $i5-4210U$) for experiments. First, MATLAB $R2016b$ was used to pre-process the data set and generate community behaviour profiles per session slots of 4 hours. Second, we implemented E-RAIDS-MCOD and E-RAIDS-AnyOut and carried out the experiments in Java environment (Eclipse Java Mars) using the open-source **MOA** package [32].

4.1 Description of the Dataset

A significant impediment to researchers who work on the insider threat problem is the lack of real world data. The real data logs the activities executed by the users in an organisation. These data log files contain: private user profile information (e.g. name, email address, mobile number, home address, etc.); intellectual property (e.g.

strategic or business plans, engineering or scientific information, source code, etc.); and confidential content (e.g. email content, file content, etc.) [7]. Organisations commonly refuse to give researchers access to real data to protect its users and assets.

In the current decade, there exists a great recent trend towards the utilisation of the CMU-CERT data set(s) for the insider threat detection systems [5, 11, 12, 29, 41, 43]. The CMU-CERT data sets are synthetic insider threat data sets generated by the CERT Division at Carnegie Mellon University [8, 15]. CMU-CERT data repository is the only one available for insider threat scenarios (5 scenarios) and has recently become the evaluation data repository for researchers addressing the insider threat problem [5, 16, 41].

In the preliminary version [21], we used $r5.1$ CMU-CERT data set, where each community consists of one malicious insider threat which map to one scenario.

For this chapter, we used $r5.2$ CMU-CERT data set which logs the behaviour of 2000 employees over 18 months. Unlike the previously released data sets, the communities in the $r5.2$ data set consist of multiple malicious insider threats which map to different scenarios. Among those employees, we extracted the data logs for employees belonging to the following three communities: Production line worker com-P, Salesman com-S, and IT admin com-I. The community com-P consists of 300 employees, 17 malicious insider threats (associated to 366 anomalous behaviours), where each threat maps to one of the scenarios $\{s1, s2, s4\}$. The community com-S consists of 298 employees, and 22 malicious insider threats (associated to 515 anomalous behaviours), where each threat maps to one of the scenarios $\{s1, s2, s4\}$. The community com-I consists of 80 employees, and 12 malicious insider threats (associated to 132 anomalous behaviours), where each threat maps to one of the scenarios $\{s2, s3\}$.

4.2 Experimental Tuning

In this work, we define two experiments based on the proposed E-RAIDS approach. The E-RAIDS-MCOD learns an ensemble of p MCOD base learners, such that each MCOD base learner is trained on a feature subspace of the p feature subspaces. Likewise, the E-RAIDS-AnyOut learns an ensemble of p AnyOut base learners.

The experiments are tuned for different values of parameters. Table 1 presents the values for the parameters tuned in each of MCOD and AnyOut, with a short description. Note that an extensive number of experiments was done to select the presented tuning values for the parameters. The values were selected based on E-RAIDS achieving the best performance in terms of the evaluation measures described below.

For MCOD, the parameter k has a default value k=50 in the MOA package. In this chapter, we present $k=\{50, 60, 70\}$ to evaluate the E-RAIDS-MCOD for different number of neighbours. The parameter r, with a default value r=0.1, is presented for a range for $r=\{0.3, 0.4, 0.5, 0.6, 0.7\}$ to check the influence of r.

Table 1: Tuned Parameters.

MCOD parameter	description
k=$\{50,60,70\}$	number of neighbours parameter
r=$\{0.3,0.4,0.5,0.6,0.7\}$	distance parameter for outlier detection
AnyOut parameter	**description**
$\|D_{train}\|$=500	training set size
$confAgr$=2	size of confidence aggregate
$conf$=4	initial confidence value
τ=$\{0.1,0.4,0.7\}$	outlier score threshold
$oscAgr$=$\{2,4,6,8\}$	size of outlier score aggregate

For AnyOut, the parameter $|D_{train}|$ is presented for 500 instances, knowing that no threats are present at the beginning of the stream in these 500 instances. The parameters $confAgr$ and $conf$ did not show a significant influence on the utilised data sets, so both are assigned to their default values, in the MOA package, 2 and 4 respectively. τ has a minimum value 0 and a maximum value 1, so it is presented for τ=$\{0.1,0.4,0.7\}$ to evaluate the influence of varying the outlier score threshold on the outliers detected. $oscAgr$ has a minimum value 1 and a maximum value 10, so it is presented for the $oscAgr$=$\{2,4,6,8\}$.

In general, for E-RAIDS approach with either MCOD or AnyOut base learner, we present the vouch factor vf_s=2, so it confirms an outlier as positive (anomalous) after it survives for 2 subsequent windows. We present the vote factor vf_e=1, so if at least 1 feature subspace in the ensemble flags an alert, an alarm of a malicious insider threat is confirmed to be generated.

Likewise, for both E-RAIDS-MCOD and E-RAIDS-AnyOut, the window size is presented for $w = \{50,100,150,200\}$. As previously described, an instance (feature vector) is a set of events executed during a pre-defined session slot. In this work, we define a session slot per 4 hours. Let w represent the window size, which accumulates the acquired instances in a data stream. If w=50 and vf_s=2, then the instances in each window are processed after 4×50=200 hours $\simeq$ 8 days. Based on the description of E-RAIDS, a threat (outlier) may be confirmed as an outlier at least over the window it belongs to (after 8 days) or over the next window (after $\simeq 8 \times 2$=16 days). If w=200 and vf_s=2, then the instances in each window are processed after 4×200=800 hours $\simeq$ 33 days. A threat (outlier) may be confirmed as an outlier at least after 33 days) or over the next window (after $\simeq 33 \times 2$=66 days). Note that the malicious insider threats simulated in the CMU-CERT data sets span over at least one month and up to 4 months. Hence, we hypothesise the following in Hypothesis 1.

Hypothesis 1: *The E-RAIDS approach is capable to detect the malicious insider threats in real-time (during the time span of the undergoing threat).*

4.3 Evaluation Measures

Many research work has been done to detect or mitigate malicious insider threats, but nevertheless has established standard measures to evaluate the proposed models [17]. The research practices show that the the insider threat problem demands to measure the effectiveness of the models before being deployed, preferably in terms of true positives (TP) and false positives (FP) [18].

In the state-of-the-art, a remarkable number of approaches were validated in terms of FP measure [4, 5, 13, 31, 35]. This sheds light on the importance of the FP measure to address the shortcoming of the high number of false alarms (FPs). Furthermore, some approaches were validated in terms of: TP measure [31]; F1 measure [4, 33]; AUC measure [9, 11, 16]; precision and recall [24, 29]; accuracy [24, 35]; and others.

The variety of the utilised evaluation measures in the state-of-the-art reveals the critical need to formulate the insider threat problem and to define the measures that would best validate the effectiveness of the propose E-RAIDS approach. In the following, we give a formulation for the E-RAIDS approach and we define the evaluation measures utilised in this work.

As aforementioned, the ultimate aim of the E-RAIDS approach is to detect all the malicious insider threats over a data stream in real-time, while minimising the number of false alarms.

The challenge of the insider threat problem lies in the variety and complexity of the malicious insider threats in the data sets. Each malicious insider threat is devised of a set of anomalous behaviours. An anomalous behaviour is represented by an instance (feature vector) which describes a set of events (features) carried out by a malicious insider. Based on the challenge of the problem, we formulate the E-RAIDS approach with the aim to detect *any-behaviour-all-threat* (as defined in Section 1). This means that it is sufficient to detect any anomalous behaviour (instance) in all malicious insider threats. Hence, E-RAIDS approach is formulated as a *threat hunting* approach, where a threat is detected if (1) a vf_e number of feature subspaces confirm an undergoing threat (flag alarm) over a window and (2) the outliers, associated to the alarm flagged over the window, include at least a True Positive (anomalous behaviour detected as outlier). Note that although the detection of one behaviour confirms the detection of the threat, we hypothesise the following in Hypothesis 2.

Hypothesis 2: *The E-RAIDS approach is capable of detecting more than one behaviour from the set of behaviours which belong to a malicious insider threat. We refer to as (more than one)-behaviour-all-threat detection in E-RAIDS.*

Furthermore, the property of the E-RAIDS approach of using windows over data streams introduces a refined version of the evaluation of False Positives (FP). In this work, we use the FPAlarm to evaluate the false positives. So that if all the outliers associated to the alarm generated over a window are actually normal, then the alarm is considered a False Alarm (i.e. FPAlarm is incremented 1). A formal definition for FPAlarm is given in the following.

We define the measures used to evaluate the performance of E-RAIDS, which include a refined version of the default (known) evaluation measures, taking our ultimate aim into account.

- P$_T$: *Threats* number of malicious insider threats associated to anomalous instances. In other words, P$_T$ is the number of malicious insiders attributed to the anomalous behaviours;
- **TP**$_T$: *True Positives* a refined version of default TP to evaluate the number of threats detected by the framework among all the P$_T$ malicious insider threats. TP$_T$ is incremented if at least one anomalous instance (behaviour attributed to the threat) is associated as an outlier to a flagged alarm;
- **FPAlarm**: *False Positive Alarm* a refined version of default FP to evaluate the number of false alarms generated. An alarm is declared false if all the outliers associated to the alarm are actually normal instances. This means that none of the instances contributed to generating the alarm, therefore, false alarm;
- FN$_T$: *False Negatives* a refined version of default FN to evaluate the number of insider threats not detected; and
- **F1** measure: defined based on the values of the above defined measures.

The evaluation for E-RAIDS does not employ only the defined evaluation measures, however, it is required to prove the previously stated hypothesises to be true.

4.4 Results and Discussion

In the preliminary version [21], RandSubOut was evaluated in terms of TP$_T$ detected threats and FPAlarm.

In this work, the results are presented and discussed for E-RAIDS-MCOD and E-RAIDS-AnyOut as follows: in terms of (1) the pre-defined evaluation measures; (2) voting feature subspaces; (3) real time anomaly detection; and (4) the detection of (more than One)-behaviour-all-threat.

4.4.1 MCOD vs AnyOut Base Learner for E-RAIDS in terms of Evaluation Measures

In the following, we analyse the performance of E-RAIDS with MCOD base learner vs AnyOut base learner in terms of the pre-defined evaluation measures: TP_T out of P_T; FPAlarm; and F1 measure.

Tables 2 and 3 present the maximum TP_T and the minimum FPAlarm attained E-RAIDS-MCOD and E-RAIDS-AnyOut over the communities. The results are reported in terms of the parameter values in the given sequence k, r, w for E-RAIDS-MCOD and $\tau, oscAgr, w$ for E-RAIDS-AnyOut respectively.

Table 2: Maximum TP_T of Detected Insider Threats over Communities.

community	E-RAIDS-MCOD	Parameters k, r, w
com-P	**16**	50,0.3,100 60,0.4,50 60,0.6,100 60,0.7,150 70,0.4,50
com-S	**21**	70,0.4,150
com-I	10	50,0.4,50 70,0.3,100

community	E-RAIDS-AnyOut	Parameters $\tau, oscAgr, w$
com-P	**16**	0.1,2,100-200 $\{0.3, 0.7\}$,2,50-200
com-S	20	0.1,2,50-100 0.3,2,50-100 0.7,2,150
com-I	**12**	0.1,2,50-200 0.3,2,50-100 0.7,2,$\{50, 200\}$

Table 3: Minimum FPAlarm over Communities.

community	E-RAIDS-MCOD	Parameters k, r, w
com-P	**0**	50,0.4,200 60,$\{0.3, 0.5, 0.6\}$,200
com-S	**0**	50,0.4,200 50,0.7,100 60,0.3,200 60,0.5-0.7,100 70,0.6-0.7,150
com-I	**0**	50,0.3,200 60,0.5,200 70,0.5-0.7,200

community	E-RAIDS-AnyOut	Parameters $\tau, oscAgr, w$
com-P	2	$\forall\tau$,$\forall oscAgr$,200
com-S	2	$\forall\tau$,$\forall oscAgr$,150-200
com-I	1	0.1,$\{2, 4\}$,200 0.3,2-6,200 0.7,2-4,200

E-RAIDS-MCOD

Fig. 3 presents the variation of $F1$ measure as a function of window size w for E-RAIDS with MCOD base learner over the communities. The results are reported with respect to k and r parameter values.

A preliminary analysis of the F1 measure shows no evident pattern in terms of any of the parameters k, r, or w. Over the community com-P, E-RAIDS-MCOD achieves the maximum F1=0.9411; $60, 0.7, 150$. It detects the maximum TP$_T$=16 out of P$_T$=17, thus missing one malicious insider threat. However, it flags only a false positive alarm (FPAlarm=1). Furthermore, Table 3 shows that E-RAIDS-MCOD reports the minimum FPAlarm=0 at w=200 for different values of k and r.

Over the community com-S, E-RAIDS-MCOD achieves the maximum F1=0.9523; $70, \{0.6, 0.7\}, 150$. It detects a TP$_T$=20 out of P$_T$=22, while flagging no false positive alarms (FPAlarm= 0). The maximum TP$_T$=21 is attained at $70, 0.4, 150$, however, flagging FPAlarm=2.

Over the community com-I, E-RAIDS-MCOD achieves the maximum F1=0.6451; $70, 0.3, 100$. It detects a TP$_T$=10 out of P$_T$=12, thus missing two malicious insider threats, while flagging FPAlarm=9. Nevertheless, Table 3 shows that E-RAIDS-

MCOD reports the minimum FPAlarm=0 at *w*=200 for different values of *k* and *r*.

We can deduce that the window size w=$150,200$ give the best performance for E-RAIDS-MCOD in terms of the evaluation measures.

E-RAIDS-AnyOut

Fig. 4 presents the variation of F1 measure as a function of window size *w* for E-RAIDS with AnyOut base learner over the communities. The results are reported with respect to τ and *oscAgr* parameter values.

It is evident that there exists a positive correlation between F1 measure and the parameter *oscAgr* for E-RAIDS-AnyOut. The variation of F1 measure at *oscAgr*=2 is the highest with respect to all the window sizes *w*=50 to 200 over both communities. Moreover, Figure 5a reveals a positive correlation between F1 measure and the parameter *w*. The value of F1 measure increases as the window size *w* increases.

Over the community com-P, E-RAIDS-AnyOut achieves the maximum F1=0.9142; $\forall\tau,2,200$. It detects the maximum TP$_T$=16 out of P$_T$=17, while reducing the false positive alarms to FPAlarm=2. Table 3 shows that E-RAIDS-AnyOut reports the minimum FPAlarm=2 $\forall\tau,\forall oscAgr$ at *w*=200. Thus, the higher the window size, the lower the FPAlarm. Table 2 shows that E-RAIDS-AnyOut reports the maximum TP$_T$=16 at *oscAgr*=2 in general terms $\forall\tau,\forall w$. Thus, the lower the *oscAgr*, the higher the TP$_T$ detected.

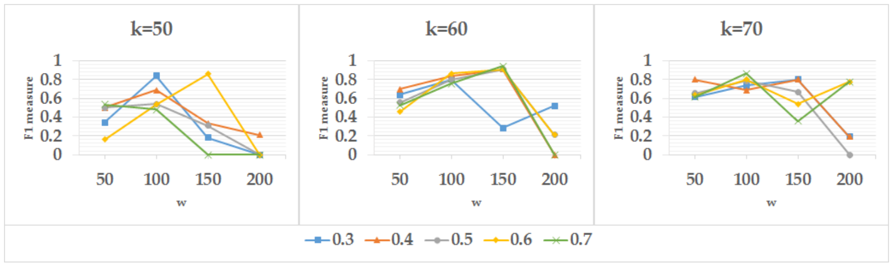
Over the community com-S, E-RAIDS-AnyOut achieves the maximum F1=0.9090; $0.7,2,150$. It detects a TP$_T$=20 out of P$_T$=22, while flagging two false positive alarms (FPAlarm=2).

Over the community com-I, E-RAIDS-AnyOut achieves the maximum F1=0.9600; $\forall\tau,2,200$. It detects a TP$_T$=12 out of P$_T$=12, while flagging one false positive alarm (FPAlarm=1).

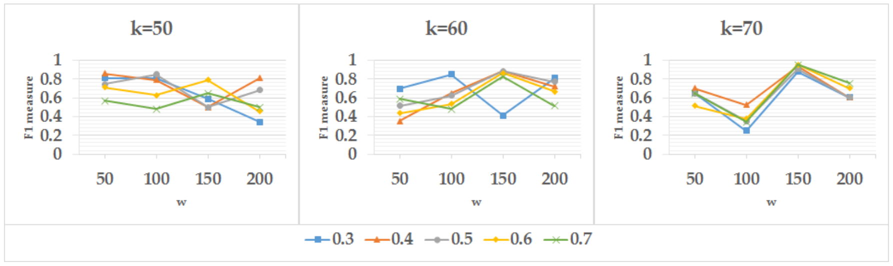
In terms of the evaluation measures, E-RAIDS-MCOD outperforms E-RAIDS-AnyOut over the communities, where E-RAIDS-MCOD achieves a higher F1 measure over com-P and com-S, a higher TP$_T$ over com-S, and a lower FPAlarm=0 over all communities.

4.4.2 MCOD vs AnyOut for E-RAIDS in terms of Voting Feature Subspaces

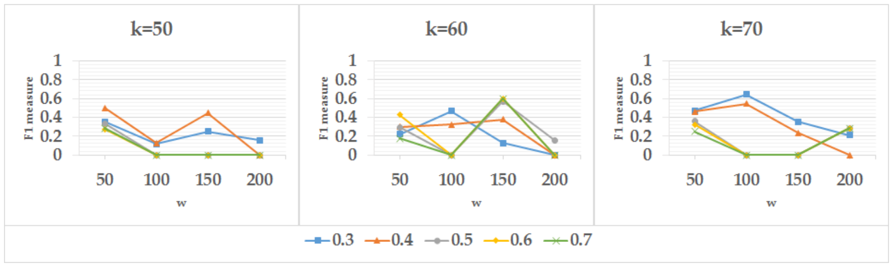
In the following, we address the merit of using feature subspaces in the E-RAIDS approach. We compare the number of feature subspaces in the ensemble that voted for a malicious insider threat in each of E-RAIDS-MCOD and E-RAIDS-AnyOut. The rationale behind using a random feature subspace to train each model of the *p* models in the ensemble, is to train the base learner (MCOD or AnyOut) on a subset of the features and to detect local outliers which might not be detected over the whole feature space.



(a) com-P.



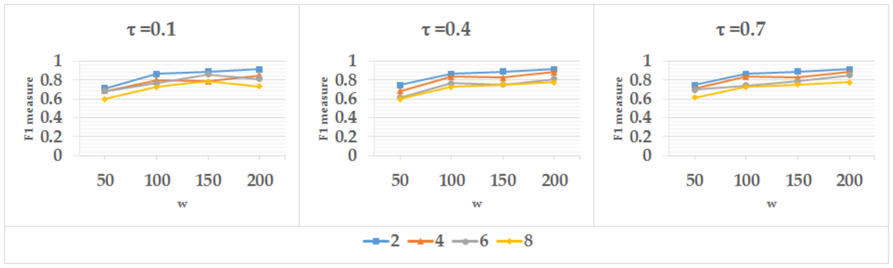
(b) com-S.



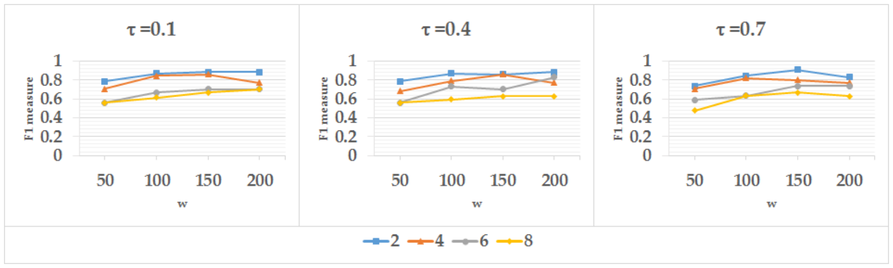
(c) com-I.

Fig. 3: The variation of F1 measure as a function of window size *w* for E-RAIDS with MCOD base learner over the communities. The legend represents the *r* parameter values.

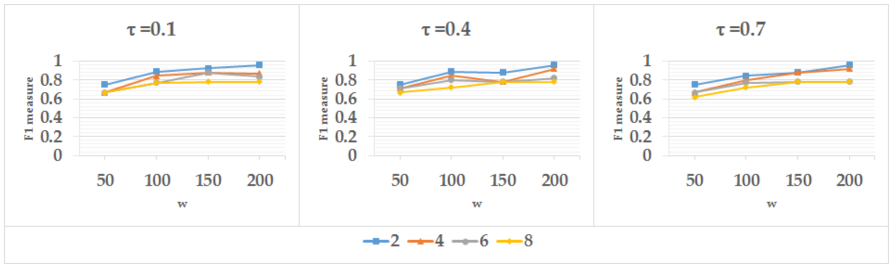
Figure 5 illustrates the number of votes which contributed in flagging an alarm of a malicious threat with respect to the number of instances processed (given the window size *w*) over the communities. The number of votes actually corresponds to the number of feature subspaces in the ensemble which generated an alert. We selected E-RAIDS-MCOD and E-RAIDS-AnyOut with their parameters which showed the best performance in terms of the evaluation measures. For com-P, Figure 5a shows the E-RAIDS-MCOD at $60, 0.7, 150$ and E-RAIDS-AnyOut at $0.1, 2, 200$. For com-S, Figure 5b shows the E-RAIDS-MCOD at $70, 0.6, 150$ and E-RAIDS-AnyOut at $0.7, 2, 150$. For com-I, Figure 5c shows the E-RAIDS-MCOD at $70, 0.3, 100$ and E-RAIDS-AnyOut at $0.1, 2, 200$. Given that the $|D_{train}|$=500, this justifies why the



(a) com-P.



(b) com-S.



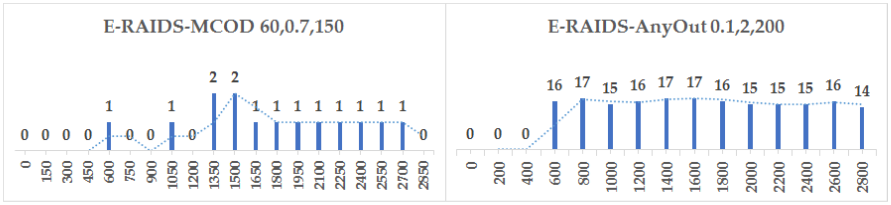
(c) com-I.

Fig. 4: The variation of F1 measure as a function of window size *w* for E-RAIDS with AnyOut base learner over the communities. The legend represents the *oscAgr* parameter values.

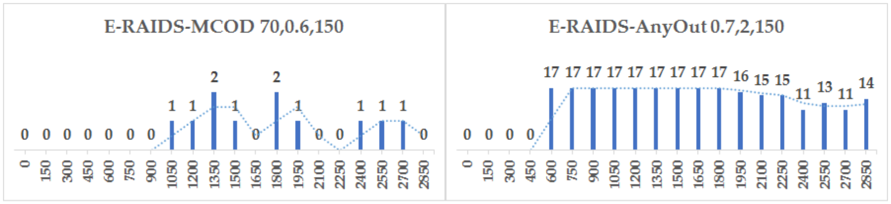
votes for E-RAIDS-AnyOut start after 500 train instances has been processed over both communities.

Given the window size *w*, a window iteration $wIter$ starts at the number of already processed instances *procInst* and ends at *procInst*+*w*. For example, given *w*=100, the first window iteration $wIter$=0 starts at *procInst*=0 and ends at *procInst* + *w*=100; $wIter$=1 starts at 100 and ends at 200; etc.

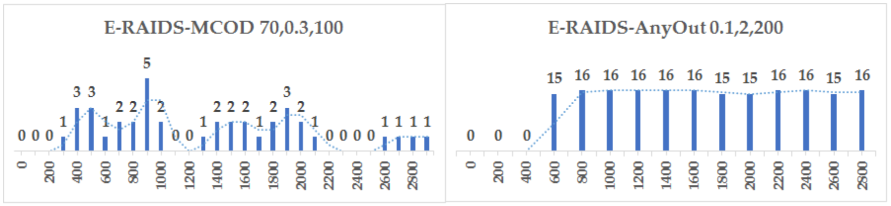
A preliminary analysis of the results shows that E-RAIDS-AnyOut flags alarms continuously $\forall wIter$ after *procInst*=500. However, E-RAIDS-MCOD shows distinct alarms flagged, with no alarms at certain windows iterations. We recall that E-RAIDS-MCOD outperforms E-RAIDS-AnyOut in terms of FPAlarm measure. The continuous alarms flagged $\forall wIter$ in E-RAIDS-AnyOut explains the higher



(a) com-P.



(b) com-S.



(c) com-I.

Fig. 5: The number of votes which contributed in flagging an alarm of a malicious threat with respect to the number of instances processed over the communities.

FPAlarm, as well as it reveals the uncertainty of E-RAIDS-AnyOut compared to E-RAIDS-MCOD.

Knowing that the number of feature subspaces utilised in the ensemble is $p = 17$, the number of votes $\forall wIter$ in E-RAIDS-AnyOut has a minimum $votes$=11 and a maximum $votes$=17, which reveals a level of uncertainty. The case where 17 feature subspaces vote for an alarm, indicates that all (17) models of the ensemble detect *at least one* outlier (positive) associated to a malicious insider threat. On the other hand, the number of votes in E-RAIDS-MCOD has a maximum $votes$=2. This means that only 1 or 2 feature subspaces vote for an alarm. We recall the complexity of the malicious insider threat scenarios in the CMU-CERT data sets. The anomalous instances usually exist in (sparse or dense) regions of normal instances, and rarely as *global outliers* with respect to the whole feature space. To address this, the E-RAIDS approach aims to detect *local outliers* which may be found over ANY (not ALL) of feature subspaces. Having all the feature subspaces in E-RAIDS-AnyOut voting for a threat, compared to a couple (1 or 2) of feature subspaces in E-RAIDS-MCOD, reinforces the uncertainty of E-RAIDS-AnyOut. The reason behind the uncertain performance of AnyOut in the E-RAIDS approach may be due to that the outlier score of an instance X^t is computed upon the arrival of a new instance X^{t+1}. Thus, the processing of the instance X^t is interrupted at a certain level of the ClusTree, and the outlier score is computed with a lower level of confidence (i.e. uncertain).

4.4.3 Real Time Anomaly Detection in E-RAIDS

To prove the aforementioned Hypothesis 1 to be true, it is required to check if the E-RAIDS detects the malicious insider threats in real-time (where real-time means that the alarm is flagged during the time span of the undergoing threat). Based on the previous conclusion regarding the uncertainty of E-RAIDS-AnyOut, and the superiority of E-RAIDS-MCOD in terms of (1) the evaluation measures and (2) the voting feature subspaces, we select E-RAIDS-MCOD to verify Hypothesis 1.

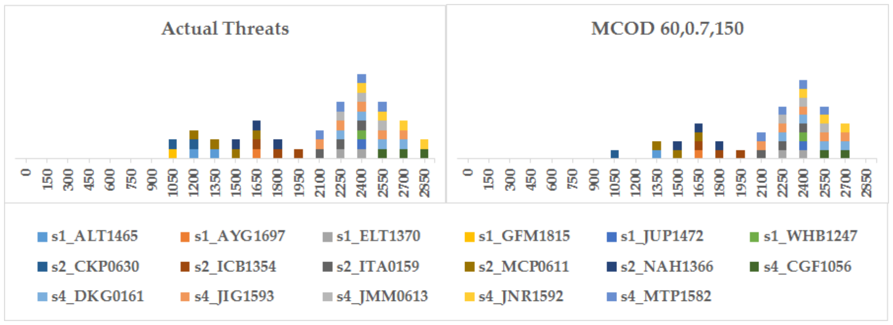
Fig. 6 illustrates the actual malicious insider threats vs the threats detected in E-RAIDS-MCOD with respect to the number of instances processed over the communities. The malicious insider threats are displayed in the legend over each community using the following label scenRef_insiderID (e.g. s1_ALT1465) such that, scenRef (e.g. s1, s2, or s4) represents the reference number for the scenario followed in the malicious insider threat; and insiderID (e.g. ALT1465, AYG1697) represents the user ID of the insider attributed to the threat. Hence, each colour in the legend refers to a malicious insider threat.

We observe in Fig. 6 that a malicious insider threat is detected either (obs1) at the current window iteration $wIter$ where it is actually simulated, or (obs2) at the subsequent (next) $wIter$. Hence, Hypothesis 1 is verified.

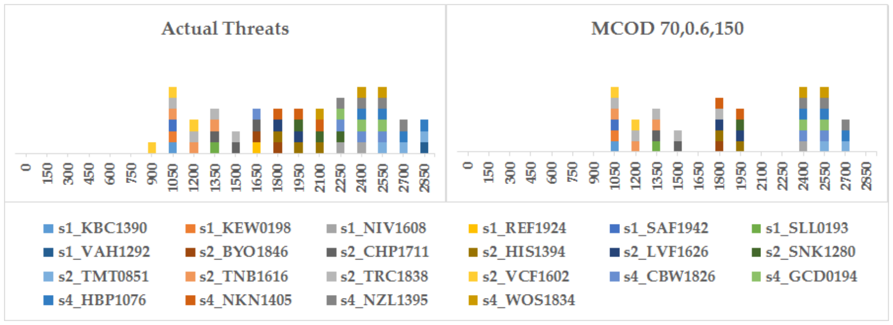
Based on the description of the E-RAIDS approach, a feature subspace votes for a threat at $wIter$ if at least an outlier survived for a vf_s number of subsequent windows, and consequently a specific threat is detected at $wIter$ if (cond1) a vf_e number of subspaces vote (an alarm is flagged), and (cond2) at least outlier (positive) associated to the alarm belongs to the threat. However, the outliers associated to the alarm (as mentioned in (cond2)) consist of *persistent* outliers (which survived from $wIter$-1) and *potential* outliers (at the current $wIter$). A potential outlier, if satisfies (cond2), allows real-time detection at the current $wIter$ (obs1). A persistent outlier, if satisfies condition (cond2), allows real-time detection as observed at the subsequent $wIter$ (obs2).

4.4.4 (More than One)-Behaviour-All-Threat Detection in E-RAIDS

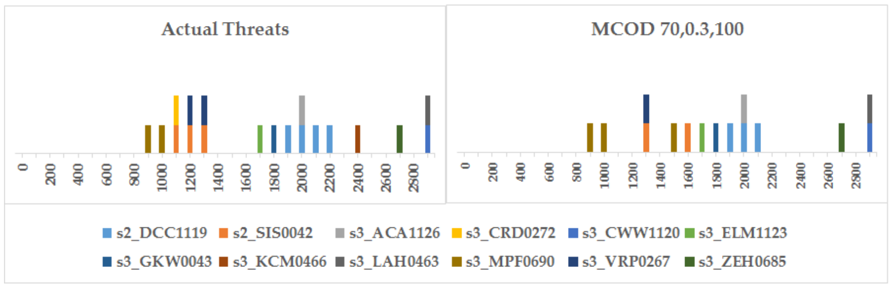
The final analysis addresses Hypothesis 2. As defined in Section 1, the idea of *threat hunting* aims to detect any-behaviour-all-threat, however, Fig. 6 shows the capability of E-RAIDS-MCOD to detect more than one behaviour (not only one) from the set of behaviours which belong to a malicious insider threat. It manifests as multiple alarms (colour spikes) generated for a specific threat over a number of windows. Hence, Hypothesis 2 is verified.



(a) com-P.



(b) com-S.



(c) com-I.

Fig. 6: The actual malicious insider threats vs the threats detected in E-RAIDS-MCOD with respect to the number of instances processed over the communities.

Analytically, this underlies in having multiple outliers, associated to the alarm(s) flagged over window(s), which are actually true positives belonging to a specific malicious insider threat.

5 Conclusion and Future Work

This chapter addresses the shortcoming of high number of false alarms in the existing insider threat detection mechanisms. The continuous flagging of false alarms deceives the administrator(s) about suspicious behaviour of many users. This consumes a valuable time from their schedule, while investigating the suspected users.

We present a streaming anomaly detection approach, namely Ensemble of Random subspace Anomaly detectors In Data Streams (E-RAIDS), for insider threat detection. The ultimate aim of E-RAIDS is to detect any-behaviour-all-threat (threat hunting as defined in section 1), while reducing the number of false alarms.

E-RAIDS is built on the top of established continuous outlier detection techniques (MCOD or AnyOut). These techniques use *data stream clustering* to optimise the detection of outliers (which may refer to malicious insider threats). E-RAIDS is an ensemble of p outlier detectors (p MCOD base learners or p AnyOut base learners), where each model of the p models learns on a random feature subspace. The merit of using feature subspaces is to detect local outliers which might not be detected over the whole feature space. These outliers may refer to anomalous behaviour(s) which belong to a malicious insider threat. E-RAIDS presents also an aggregate component to combine the votes from the feature subspaces, and take a decision whether to flag an alarm or not.

We define two experiments: E-RAIDS-MCOD with MCOD base learner, and E-RAIDS-AnyOut with AnyOut base learner. The experiments are carried out on CMU-CERT data sets which include simulated malicious insider threats scenarios. We compare the performance of E-RAIDS using each of MCOD and AnyOut in terms of, (1) the evaluation measures: F1 measure, TP_t of threats detected, and FPAlarm flagged; (2) the effectiveness of the concept of voting feature subspaces; (3) the capability of E-RAIDS to detect insider threats in real-time (Hypothesis 1); and (4) the capability of E-RAIDS to detect more than one behaviour belonging to an insider threat (Hypothesis 2) despite our formulation to the insider threat approach (threat hunting).

The results show that E-RAIDS-MCOD outperforms E-RAIDS-AnyOut, where the latter shows a low level of certainty in the detection of outliers. E-RAIDS-MCOD reports a higher F1 measure=0.9411 and 0.9523 over com-P and com-S, a lower FPAlarm=0 over all communities, and misses only one threat TP_T=16 and 21 over com-P and com-S. It is worth to also mention that the window size w=$150, 200$ gives the best performance for E-RAIDS-MCOD compared to the tuned values. E-RAIDS verifies the hypothesised capabilities in terms of the detection of more than one behaviour per threat in real-time.

References

1. Aggarwal, C.C., Philip, S.Y., Han, J., Wang, J.: -a framework for clustering evolving data streams. In: Proceedings 2003 VLDB Conference, pp. 81–92. Elsevier (2003)

2. Angiulli, F., Fassetti, F.: Distance-based outlier queries in data streams: the novel task and algorithms. Data Mining and Knowledge Discovery **20**(2), 290–324 (2010)
3. Assent, I., Kranen, P., Baldauf, C., Seidl, T.: Anyout: Anytime outlier detection on streaming data. In: International Conference on Database Systems for Advanced Applications, pp. 228–242. Springer (2012)
4. Azaria, A., Richardson, A., Kraus, S., Subrahmanian, V.: Behavioral analysis of insider threat: a survey and bootstrapped prediction in imbalanced data. IEEE Transactions on Computational Social Systems **1**(2), 135–155 (2014)
5. Böse, B., Avasarala, B., Tirthapura, S., Chung, Y.Y., Steiner, D.: Detecting insider threats using radish: A system for real-time anomaly detection in heterogeneous data streams. IEEE Systems Journal (2017)
6. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. In: ACM sigmod record, vol. 29, pp. 93–104. ACM (2000)
7. Cappelli, D.M., Moore, A.P., Trzeciak, R.F.: The CERT guide to insider threats: how to prevent, detect, and respond to information technology crimes (Theft, Sabotage, Fraud). Addison-Wesley (2012)
8. CERT Team, C.M.U.: Cmu cert synthetic insider threat data set. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099. [Online; accessed 12-April-2018]
9. Chen, Y., Nyemba, S., Malin, B.: Detecting anomalous insiders in collaborative information systems. IEEE transactions on dependable and secure computing **9**(3), 332–344 (2012)
10. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd, vol. 96, pp. 226–231 (1996)
11. Gamachchi, A., Boztas, S.: Insider threat detection through attributed graph clustering. In: Trustcom/BigDataSE/ICESS, 2017 IEEE, pp. 112–119. IEEE (2017)
12. Gamachchi, A., Sun, L., Boztas, S.: Graph based framework for malicious insider threat detection. In: Proceedings of the 50th Hawaii International Conference on System Sciences, pp. 2638,2647 (2017)
13. Gates, C., Li, N., Xu, Z., Chari, S.N., Molloy, I., Park, Y.: Detecting insider information theft using features from file access logs. In: European Symposium on Research in Computer Security, pp. 383–400. Springer (2014)
14. Georgiadis, D., Kontaki, M., Gounaris, A., Papadopoulos, A.N., Tsichlas, K., Manolopoulos, Y.: Continuous outlier detection in data streams: an extensible framework and state-of-the-art algorithms. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, pp. 1061–1064. ACM (2013)
15. Glasser, J., Lindauer, B.: Bridging the gap: A pragmatic approach to generating insider threat data. In: Security and Privacy Workshops (SPW), 2013 IEEE, pp. 98–104. IEEE (2013)
16. Goldberg, H., Young, W., Reardon, M., Phillips, B., et al.: Insider threat detection in prodigal. In: Proceedings of the 50th Hawaii International Conference on System Sciences (2017)
17. Greitzer, F.L., Ferryman, T.A.: Methods and metrics for evaluating analytic insider threat tools. In: Security and Privacy Workshops (SPW), 2013 IEEE, pp. 90–97. IEEE (2013)
18. Guido, M.D., Brooks, M.W.: Insider threat program best practices. In: System Sciences (HICSS), 2013 46th Hawaii International Conference on, pp. 1831–1839. IEEE (2013)
19. Guttman, A.: R-trees: A dynamic index structure for spatial searching, vol. 14. ACM (1984)
20. Hahsler, M., Bolanos, M., Forrest, J., et al.: Introduction to stream: An extensible framework for data stream clustering research with r. Journal of Statistical Software **76**(14), 1–50 (2017)
21. Haidar, D., Gaber, M.M.: Outlier detection in random subspaces over data streams: An approach for insider threat detection. Expert Update **17**(1), 1–16 (2017)
22. Hawkins, D.M.: Identification of outliers, vol. 11. Springer (1980)
23. He, Z., Xu, X., Deng, S.: Discovering cluster-based local outliers. Pattern Recognition Letters **24**(9-10), 1641–1650 (2003)
24. Kandias, M., Stavrou, V., Bozovic, N., Gritzalis, D.: Proactive insider threat detection through social media: The youtube case. In: Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society, pp. 261–266. ACM (2013)

25. Khalilian, M., Mustapha, N.: Data stream clustering: Challenges and issues. arXiv preprint arXiv:1006.5261 (2010)
26. Knox, E.M., Ng, R.T.: Algorithms for mining distancebased outliers in large datasets. In: Proceedings of the International Conference on Very Large Data Bases, pp. 392–403. Citeseer (1998)
27. Kontaki, M., Gounaris, A., Papadopoulos, A.N., Tsichlas, K., Manolopoulos, Y.: Continuous monitoring of distance-based outliers over data streams. In: 2011 IEEE 27th International Conference on Data Engineering, pp. 135–146. IEEE (2011)
28. Kranen, P., Assent, I., Baldauf, C., Seidl, T.: Self-adaptive anytime stream clustering. In: Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on, pp. 249–258. IEEE (2009)
29. Legg, P.A., Buckley, O., Goldsmith, M., Creese, S.: Automated insider threat detection system using user and role-based profile assessment. IEEE Systems Journal (2015)
30. Legg, P.A., Buckley, O., Goldsmith, M., Creese, S.: Caught in the act of an insider attack: Detection and assessment of insider threat. In: Technologies for Homeland Security (HST), 2015 IEEE International Symposium on, pp. 1–6. IEEE (2015)
31. Mayhew, M., Atighetchi, M., Adler, A., Greenstadt, R.: Use of machine learning in big data analytics for insider threat detection. In: Military Communications Conference, MILCOM 2015-2015 IEEE, pp. 915–922. IEEE (2015)
32. MOA Team, T.U.o.W.: Massive online analysis open source framework. https://moa.cms.waikato.ac.nz/. [Online; accessed 14-February-2018]
33. Moriano, P., Pendleton, J., Rich, S., Camp, L.J.: Insider threat event detection in user-system interactions. In: Proceedings of the 2017 International Workshop on Managing Insider Security Threats, pp. 1–12. ACM (2017)
34. Nurse, J.R., Legg, P.A., Buckley, O., Agrafiotis, I., Wright, G., Whitty, M., Upton, D., Goldsmith, M., Creese, S.: A critical reflection on the threat from human insiders–its nature, industry perceptions, and detection approaches. In: International Conference on Human Aspects of Information Security, Privacy, and Trust, pp. 270–281. Springer (2014)
35. Parveen, P., Mcdaniel, N., Weger, Z., Evans, J., Thuraisingham, B., Hamlen, K., Khan, L.: Evolving insider threat detection stream mining perspective. International Journal on Artificial Intelligence Tools **22**(05), 1360,013 (2013)
36. Parveen, P., Weger, Z.R., Thuraisingham, B., Hamlen, K., Khan, L.: Supervised learning for insider threat detection using stream mining. In: 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence, pp. 1032–1039. IEEE (2011)
37. Reuters: Harold marin theft of ip. http://www.reuters.com/article/us-usa-cybersecurity-nsa-contractor-idUSKBN15N2N4. [Online; accessed 14-February-2018]
38. Seidl, T., Assent, I., Kranen, P., Krieger, R., Herrmann, J.: Indexing density models for incremental learning and anytime classification on data streams. In: Proceedings of the 12th international conference on extending database technology: advances in database technology, pp. 311–322. ACM (2009)
39. Silva, J.A., Faria, E.R., Barros, R.C., Hruschka, E.R., De Carvalho, A.C., Gama, J.: Data stream clustering: A survey. ACM Computing Surveys (CSUR) **46**(1), 13 (2013)
40. Software Engineering Institute, C.M.U.: 2011 cybersecurity watch survey. https://www.sei.cmu.edu/news/article.cfm?assetid=52441. [Online; accessed 14-February-2018]
41. Tuor, A., Kaplan, S., Hutchinson, B., Nichols, N., Robinson, S.: Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. arXiv preprint arXiv:1710.00811 (2017)
42. Verble, J.: The nsa and edward snowden: surveillance in the 21st century. ACM SIGCAS Computers and Society **44**(3), 14–20 (2014)
43. Walton, S., Maguire, E., Chen, M.: Multiple queries with conditional attributes (qcats) for anomaly detection and visualization. In: Proceedings of the Eleventh Workshop on Visualization for Cyber Security, pp. 17–24. ACM (2014)
44. Yang, D., Rundensteiner, E.A., Ward, M.O.: Neighbor-based pattern detection for windows over streaming data. In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, pp. 529–540. ACM (2009)

45. Zargar, A., Nowroozi, A., Jalili, R.: Xaba: A zero-knowledge anomaly-based behavioral analysis method to detect insider threats. In: Information Security and Cryptology (ISCISC), 2016 13th International Iranian Society of Cryptology Conference on, pp. 26–31. IEEE (2016)
46. Zhang, T., Ramakrishnan, R., Livny, M.: Birch: an efficient data clustering method for very large databases. In: ACM Sigmod Record, vol. 25, pp. 103–114. ACM (1996)

Outlier Detection in Random Subspaces over Data Streams: An Approach for Insider Threat Detection

Diana Haidar and Mohamed Medhat Gaber

Abstract Insider threat detection is an emergent concern for industries and governments due to the growing number of attacks in recent years. Several Machine Learning (ML) approaches have been developed to detect insider threats, however, they still suffer from a high number of false alarms. None of those approaches addressed the insider threat problem from the perspective of stream mining data where a concept drift or an outlier is an indication of an insider threat. An outlier refers to anomalous behaviour that deviates from the normal baseline of community's behaviour and is the focus of this paper. To address the shortcoming of existing approaches and realise a novel solution to the problem, we present RandSubOut (Random Subspace Outliers) approach for insider threat detection over real-time data streams. RandSubOut allows the detection of insider threats represented as localised outliers in random feature subspaces, which would not be detected over the whole feature space, due to dimensionality. We evaluated the presented approach as an ensemble of established distance-based outlier detection methods, namely, Micro-cluster-based Continuous Outlier Detection (MCOD) and Anytime OUTlier detection (AnyOut), according to evaluation measures including True Positive (TP) and False Positive (FP).

1 Introduction

Insider threat detection is an emergent concern for academia, industries, and governments due to the growing number of attacks in recent years. Insiders

Diana Haidar
Birmingham City University, Birmingham, UK e-mail: diana.haidar@bcu.ac.uk

Mohamed Medhat Gaber
Birmingham City University, Birmingham, UK e-mail: mohamed.gaber@bcu.ac.uk

are current or former employees, contractors or business partners of an organisation who have authorised access to the network system, data, or sensitive information (e.g. trade secrets, organisation plans, and Intellectual Property) [15]. Their authorised access makes the organisation prone to insider threats that could have a vital negative impact on the system. Insider threats can be categorised into two groups, which are malicious insider threats and unintentional insider threats. Malicious insider threats are caused by insiders who exploit their privileges with the intention to compromise the confidentiality, integrity, or availability of the system and data. Their motivation would be to disclose or modify organisation's information for financial gain, for the advantage of a competitor company with the promise of a higher-paying job, or for revenge [4]. Unintentional insider threats are caused by insiders who accidentally harm the organisation's assets. This includes human mistakes and errors that are the result of faulty system design or the non-malicious negligence of insiders.

Legg *et al.* [14] stated that according to the 2011 Cybersecurity Watch Survey, 58% of attacks are associated outsiders, while 21% are associated insiders. However, the risk of insider attacks is much more than that of outsider attacks, because insiders have authorised access to sensitive data that could be used to damage the company's reputation, brand integrity, or customer confidence. An important example is the insider attack of Edward Snowden [21], a contractor hired by the National Security Agency (NSA). Snowden, in May 2013, stole approximately 1.7 million documents containing secret data from the NSA and revealed it to the public via newspapers and mass media. This incident was reported as the biggest US intelligence leakage, and has caused increased concern amongst governments and companies. Moreover, there exist various security measures for outsider threats, which is not the case for insider threats, such as firewalls, Intrusion Detection/Prevention Systems (IDS/IPS), and antiviruses.

The continuous streaming of insider threat data generated from various sources (e.g. security logs, traffic data, web requests, and email headers) increases the volume and velocity of this data acquisition. To handle the big data streams and detect insider threats, several machine learning approaches have been developed, however, they still suffer from a high number of false alarms. None of those approaches addressed the insider threat problem from the perspective of *stream mining* data where a *concept drift* or an *outlier* is an indication of an insider threat. We can recognize two types of drifts: a normal drift and a concept drift. A normal drift is the change in data distribution over time; the normal change in a user's behaviour over time according to tasks assigned, developed knowledge level and other elements [24]. A concept drift may manifest in different patterns such as sudden/abrupt, incremental, gradual, and reoccurring concepts [9]. The focus of this paper is on *outliers* which refer to anomalous behaviour that deviates from the normal baseline of user's behaviour or community (i.e. a group of users having the same role) behaviour.

Hence, we present RandSubOut (Random Subspace Outliers) approach for insider threat detection over real-time data streams. The idea is to construct an ensemble of $p+1$ outlier detection base learners, including p learners over random subspaces of feature pairs and one learner over all feature space. RandSubOut allows the detection of insider threats represented as localised outliers in random feature subspaces, which would not be the case over the whole feature space. The main contributions of this paper are summarised as follows:

- extraction of new feature types from the synthetic insider threat data by CMU−CERT [1] [11] to generate community behaviour profiles;
- presentation of RandSubOut for Insider Threat Detection with MCOD and AnyOut outlier detection base learners;
- detection of insider threats represented as distance-based outliers using RandSubOut; and
- evaluation of the performance of RandSubOut according to the evaluation measures including TP and FP.

The rest of this paper is organised as follows. Section 2 summarises the related work. Section 3 introduces some outlier detection methods over a data stream and describes RandSubOut for Insider Threat Detection. Experiments and results are discussed in Section 4. Finally, Section 5 concludes the paper and suggests future work.

2 Related Work

There exists a significant body of research on applying ML approaches for detecting insider threats. A great research trend is towards utilising probabilistic and k Nearest Neighbours (k-NN) approaches. The probabilistic ML approaches include Naive Bayes [25][19], Bayesian Network (BN) [3], and Hidden Markov Model (HMM) [12]. Others chose to combine BN and other ML algorithms, such as HMM [20] and Stochastic Gradient Boosting [5], with the aim to improve their models' performance. While, the k-NN approaches [7][8][18] are distance-based so any deviation of a user's behaviour from her/his normal baseline or the baseline of the community's behaviour flags anomalous behaviour. In addition, there exists research on implementing Fuzzy Inference Systems [24][6] to evaluate the degree of insider threat.

However, neither of those approaches addressed the insider threat problem from the perspective of mining data streams with concept drift, except for [16]. Parveen *et al.* [16] proposed an ensemble of One Class Support Vector Machine (OCSVM) to model a time series of daily logs. It maintains a k number of ensemble models having the minimum prediction error. The ensemble of OCSVM reported higher accuracy and almost half the number of

[1] https://www.cert.org/insider-threat/tools/

false positives compared to traditional OCSVM. The authors extended their work in a future paper [17] where the ensemble approach was applied on unsupervised Graph-based Anomaly Detection (GBAD). The results proved that the ensemble approach based on supervised OCSVM outperformed that based on unsupervised GBAD which reported a higher false rate.

Parveen's approach [16] models normal user's behaviour acquired from data streams over an ensemble of OCSVM classifiers in order to find a decision boundary between normal class label $c = 1$ and abnormal class label $c = 0$ (i.e. malicious insider threats). However, the idea of RandSubOut is to detect malicious insider threats represented as distance-based outliers continuously over real-time data streams. It also allows to detect localised outliers over random feature subspaces which may not be detected over the whole feature space as addressed in [16]. In addition, the paper [16] evaluated the proposed approach over 1998 DARPA Intrusion Detection data set [2] which is not specifically developed for insider threat detection. Unlike [16], our approach RandSubOut is evaluated over a synthetic data set with insider threat scenarios generated by CMU$-$CERT.

3 Outlier based Insider Threat Detection

This section describes an established distance-based outlier detection methods, as well as the proposed approach RandSubOut utilizing those methods for insider threat detection.

3.1 Outlier Detection Methods

The state of the art introduces different outlier detection methods over a data stream. The paper [10] provides a comparison of four distance-based outlier detection methods including STORM [1], Abstract-C [22], COD [13], and MCOD [13]. According to [10], COD and MCOD have lower space and time requirements than STORM and Abstract-C. However, all of the four outlier detection methods output the same outliers [10]. Thus, STORM and Abstract-C were excluded from being tested in the proposed approach. Moreover, the experimental evaluation in [13] shows better performance for MCOD compared to COD over benchmark tested data sets. Hence, MCOD was selected among the four methods. All of the four methods in addition to a further distance-based outlier detection method called AnyOut [2] have been implemented in the open-source tool for Massive Online Analysis (MOA)[3].

[2] http://www.ll.mit.edu/ideval/data/1998data.html

[3] http://moa.cms.waikato.ac.nz/

We select MCOD and AnyOut, which has not been compared to the four methods, to test their performance on synthetic insider threat data sets generated by the CERT Division at Carnegie Mellon University (CMU−CERT). A description of the selected algorithms is provided below.

3.1.1 MCOD

MCOD (Micro-cluster-based Continuous Outlier Detection) is an event-based approach that introduces the concept of evolving micro-clusters. The micro-clusters are the regions containing inlier instances with no overlapping. This allows to evaluate the range queries for new instance x with respect to the centers of micro-clusters instead of all the preceding active instances in the current window w. Thus, reducing the space and time requirements. Let S represent the set of instances that doesn't belong to any micro-cluster. For each new instance x, MCOD selects the center of the nearest micro-cluster c. If the $dist(x, c) \leq \frac{r}{2}$ such that r is the distance parameter for outlier detection, x is assigned to the corresponding micro-cluster. Otherwise, a range query for x is evaluated with respect to the set S of instances that doesn't belong to any micro-cluster. Let b represent the number of neighbours $N \subseteq S$ of x where $\forall n \in N, dist(x, n) \geq \frac{r}{2}$ and let k represent the number of neighbours parameter. If $b > \theta k$ such that parameter $\theta \geq 1$, then a new micro-cluster with center x is created and the neighbours N are assigned to this micro-cluster. A micro-cluster whose size decreases below $k + 1$ is deleted and a range query similar to that described for x is performed for each of its former instances. An instance x is flagged as an outlier, if there exists less than k instances in either S or in each micro-cluster of center c that satisfies the following: $dist(x, c) \leq \frac{3}{2}r$.

3.1.2 AnyOut

AnyOut (Anytime Outlier detection) is a cluster-based approach that suggests a hierarchy of clusters in tree structure named ClusTree where upper level clusters include fine grained information about lower levels. It represents a cluster by a Cluster Feature tuple $CF = (n, LS, SS)$, where n is the number of instances in the cluster, LS and SS are respectively the linear sum and the squared sum of those instances. The compact structure of the tree using CF tuples reduces space requirements. The idea of AnyOut is to traverse the tree in top-down manner and compare the current instance x_i to the clusters at each level until the arrival of new instance x_{i+1} interrupts the descent down the tree. At the moment of interruption, it inserts x_i to the cluster node e arrived into in the ClusTree and provides its degree of outlierness. Thus, maintaining real-time feedback. AnyOut defines two scores to reflect the degree of outlierness of an instance x_i. Mean outlier score is $dist(x_i, \mu(e))$

where $\mu(e)$ is the mean of e that x_i is inserted to. Density outlier score is $1 - gaus(x_i, e)$ where $gaus(x_i, e)$ is the Gaussian probability density of x_i for $\mu(e)$.

3.2 RandSubOut Approach for Insider Threat Detection

Features extracted from system and network logs describe the activity of a community. The deviation of such activity from its normal pattern according to a certain determined threshold could correlate to a malicious insider behaviour [23]. Those features are treated as *priori indicators*. For example, increased removable media activity which could correlate to IP theft; logon after working hours or from a different county location; vast number of file events like downloads, uploads, or email forwards. Another indicator is related to text categorization where specific keywords, phrases or emails could correlate to malicious insider activities. An illustrative case could be browsed suspicious websites and on the top of the list is WikiLeaks and job hunting sites.

The feature space identified in this research consists of a combination of system and network logs. The feature set assesses each user's logged activities, devices, and assigned attribute values. It is categorized into four feature types sorted according to session slots:

- *Frequency-based features* (e.g. freq. (i.e. frequency) of logon, freq. of connecting a device, freq. of copying files to removable media device, etc.);
- *Time-based features* (e.g. logon after working hours, device usage after working hours, duration of connecting a device, etc.);
- *Boolean features* (e.g. logon from new pc flag= 0,1, email-bcc non-empty, email-to or -cc or -bcc include a non-employee, etc.);
- *Attribute-based features* (e.g. freq. of visiting a particular URL, freq. of sending emails to a particular user, etc.).

We propose an approach for insider threat detection named RandSubOut (Random Subspace Outliers). The aim of RandSubOut is based on detecting localised outliers in random subspaces of feature pairs. Those localised outliers represent insider threats, which would not be detected over the whole feature space. For instance, let f represent the number of features and $p = f$ represent the number of feature subspaces. The idea is to construct an ensemble of $p+1$ base learners, including p learners over random feature subspaces and one learner over all feature space. Each learner in the ensemble will build a model, so based on a voting mechanism of those models, an alarm may be generated reporting a malicious insider threat. In the following, a detailed description of RandSubOut is provided.

3.2.1 Over each feature subspace

At each window slide $wIter$ (fixed size of window slide), RandSubOut creates an outlier temporal list $OutTempList$ of the detected outliers $OutSet$, each associated with a temporal factor $tempF$ and the window slide $wIter$. A temporal factor $tempF$ counts the number of windows that the outlier survived in (i.e. the outlier did not turn into an inlier). For example, if an outlier $out \in OutTempList$ remains as an outlier at next window slide $wIter + 1$, $tempF$ associated to out is incremeted by 1.

RandSubOut then creates a list of flag alerts $SubFlagAlertList$ associated to the window slides $wIter$ for the corresponding feature subspace. A flag alert $flagAlert$ is a boolean variable assigned to $\{0, 1\}$. We define a vouch factor $vouchF$ as a parameter which confirms whether an outlier is a positive (i.e. a malicious insider threat). At each window slide $wIter$, we check if $tempF = vouchF$, then the outlier survived for a $vouchF$ number of windows and $flagAlert$ is assigned 1 for the window slide $wIter$ that the outlier is confirmed as a positive at. Thus, minimizing the number of false alarms (i.e. false flag alerts). Algorithm 1 gives a a pseudo code to describe the steps of the procedure over each feature subspace. Note that $outMethod$ represents the distance-based outlier detection method utilized as a base learner. This refers to either MCOD or AnyOut in this paper.

3.2.2 Over the ensemble of random feature subspaces

In total, RandSubOut creates a list of votes $EnsembleVoteList$ which counts the number of times a $flagAlert$ is assigned to 1 at each window slide $wIter$ over all the random feature subspaces. This count is maintained in a variable $vote$. Let $voteF$ represent a vote factor which confirms whether a flag alarm should be generated as a total vote of the ensemble. If $vote \geq voteF$, then a an alarm is generated at this window slide $wIter$. A description for the procedure over the ensemble of random feature subspaces is provided in a pseudo code in Algorithm 2.

4 Experiments

In this section, we present the experiments performed to evaluate RandSubOut with MCOD and AnyOut based learners over CMU$-$CERT data. The results are displayed and discussed in order to select the optimal tuned parameters for detecting malicious insider threat represented as outliers.

Algorithm 1 Over each feature subspace

```
1: $wIter \leftarrow 0$
2: $found \leftarrow 0$
3: while $nbrProcInst < nbrInst$ do
4:     outMethod.process($newInst$)
5:     if ($nbrProcInst \mod w) = 0$ then
6:         $OutSet \leftarrow$ outMethod.getOutliersFound()
7:         for Outlier $out$ : $OutSet$ do
8:             if $out \in$ OutTempList.getOutliers() then
9:                 $tempF \leftarrow$ OutTempList.getOutTuple($out$).getTempF()
10:                OutTempList.getOutTuple($out$).setTempF($tempF + 1$)
11:                $found \leftarrow 1$
12:            end if
13:            if $found = 0$ then
14:                $tempF \leftarrow 1$
15:                OutTempList.addOutTuple($out$, $tempF$, $wIter$)
16:            end if
17:        end for
18:        for Outlier $out \in$ OutTempList.getOutliers() do
19:            if OutTempList.getOutTuple($out$).getTempF()$= vouchF$ then
20:                if SubFlagAlertList.getFlagAlert($wIter$)$= 0$ then
21:                    $flagAlert \leftarrow 1$
22:                    SubFlagAlertList.setFlagAlert($wIter$, $flagAlert$)
23:                end if
24:                OutTempList.removeOutTuple($out$)
25:            else
26:                $outWIter \leftarrow$ OutTempList.getOutTuple($out$).getWIter()
27:                if $wIter - outWIter + 1 = vouchF$ then
28:                    OutTempList.removeOutTuple($out$)
29:                end if
30:            end if
31:        end for
32:        $wIter \leftarrow wIter + 1$
33:    end if
34: end while
```

Algorithm 2 Over the ensemble of random feature subspaces

```
1: for FeatureSubspace $FSubspace$ : FeatureSpace do
2:     for WindowIterSlide $wIter$ : SubFlagAlertList.getWIters() do
3:         $tempVote \leftarrow$ EnsembleVoteList.getVote($wIter$)
4:         $vote \leftarrow tempVote +$ SubFlagAlertList.getFlagAlert($wIter$)
5:         EnsembleVoteList.setVote($wIter$, $vote$)
6:     end for
7: end for
```

4.1 Experimental Setup

We extracted a set of the features defined previously from the CMU−CERT data sets. This feature set builds up a community behaviour profile for each community (e.g. computer programmers, mathematicians, test engineers,etc.) sorted according to session slots. A session slot is per 4 hours to handle the detection of insider threat as soon as possible near to real-time. Since the range of feature values varies widely, we performed feature scaling in the range of $[0, 1]$.

As discussed before, a malicious insider threat may manifest in different patterns of concept drift or an outlier which is the focus of this paper. We consider a malicious insider threat scenario that maps to outliers, where a user uses a USB drive at markedly higher rate than previous activity to steal data [4] [11]. The total number of instances $nbrInst = 3000$ in a community behaviour model represents the number of session slots. The period of the malicious insider threat scenario maps to a start at $lowerBound = 922$ and an end at $upperBound = 1323$.

We evaluated RandSubOut as an ensemble of MCOD base learners and an ensemble of AnyOut base learners according to TP (i.e. true alarms that correlate to malicious insider threat) and FP (i.e. false alarms that correlate to normal instances detected as outliers) measures.

For MCOD, Table 1 displays a description of the tuned parameters. For AnyOut, the experiments revealed that varying the values of the parameters $trainSetSize$, $confAggr$, $conf$, and $threshold$ does not have a significant effect on the results, so they were assigned to fixed values. However, we tuned other parameters as described in Table 2.

Table 1 MCOD Tuned Parameters.

$k = \{50, 60, 70\}$	number of neighbours parameter
$r = \{0.3, 0.4, 0.5, 0.6, 0.7\}$	distance parameter for outlier detection
$w = \{100, 150, 200\}$	window size

Table 2 AnyOut Tuned Parameters.

$trainSetSize = 500$	training set size
$confAggr = 2$	size of confidence aggregate
$conf = 4$	initial confidence value
$threshold = 0.4$	outlier score threshold
$oScoreAggr = \{2, 8\}$	size of outlier score aggregate
$w = \{100, 200\}$	window size

[4] https://www.cert.org/insider-threat/tools/

The vouch factor $vouchF$ defined previously is assigned $vouchF = 2$ for MCOD, so it confirms whether an outlier is positive after it survives for 2 window slide iterations. However, it is tuned to $vouchF = \{2, 4\}$ for AnyOut, because it affects the results, which is not the case for MCOD where no flag alarms are generated then. The vote Factor $voteF$ is assigned $voteF = 2$, so it confirms whether a flag alarm should be generated as a total vote of the ensemble.

4.2 Results and Discussion

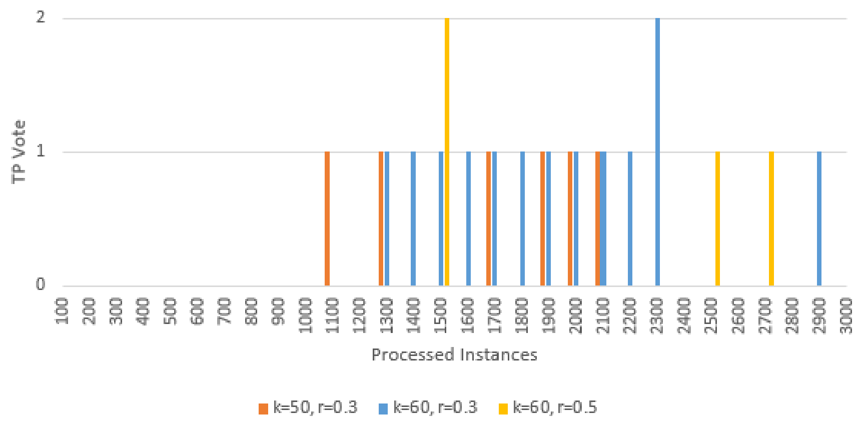


Fig. 1 TP votes generated by RandSubOut with MCOD over $w = 100$.

Fig.1, Fig.2, and Fig.3 show the number of TP votes generated by RandSubOut using MCOD base learners over window size $w = 100$, $w = 150$, and $w = 200$ respectively with varying values of parameters k and r. Note that the parameters which didn't show TP votes are not displayed in figures. The window slide iteration $wIter$ ends at the number of processed instances $nbrProcInst$ and starts at $nbrProcInst - w$ (e.g. for $w = 100$, the first window slide iteration $wIter = 1$ ends at $nbrProcInst = 100$ and starts at $nbrProcInst - w = 0$). In order to satisfy real-time or near real-time detection of a malicious insider threat (i.e. TP), we should consider the flag alarms generated at real-time or near real-time. For instance, in Fig.1, if $vouchF = 2$, then a flag alarm of TP votes should be generated between $lowerVouchBound = ceil(lowerBound + w \times (vouchF - 1)) = floor(922 + 100 \times 1) = 1100$ and $upperVouchBound = floor(upperBound + w \times vouchF) = ceil(1323 + 100 \times 1) = 1500$. Fig.1 shows that the alarms started to be flagged at $lowerVouchBound = 1100$, however more flag alarms

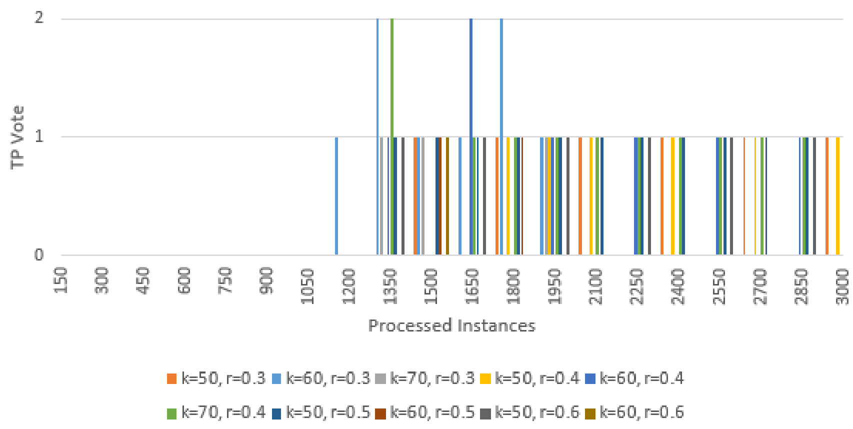


Fig. 2 TP votes generated by RandSubOut with MCOD over $w = 150$.

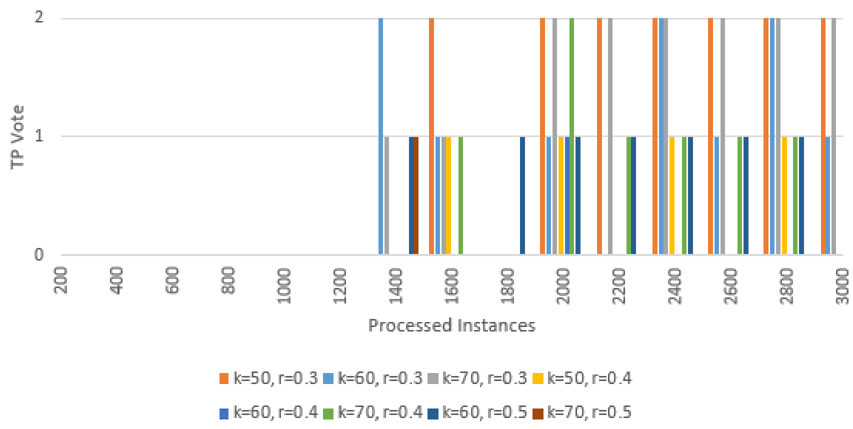


Fig. 3 TP votes generated by RandSubOut with MCOD over $w = 200$.

are generated after $upperVouchBound = 1500$. This is justified by that instances assigned as inliers by RandSubOut may turn into outliers after a number of window slide iterations. In MCOD, an instance may not belong to any micro-cluster but still is an inlier, thus considered a potential outlier. RandSubOut considers a flag alarm as a true if it signals a $vote \geq voteF$ and it is generated between $lowerVouchBound$ and $upperVouchBound$. This means that at least two feature subspaces voted for a flag alarm at this window slide iteration $wIter$. Thus, minimizing the number of false alarms. To select the optimal parameter values, we consider the parameter values which satisfies a true flag alarms ONLY generated between $lowerVouchBound$ and $upperVouchBound$. Fig. 1 for $k = 60, r = 0.5, w = 100$ reports a true alarm with $vote \geq voteF; voteF = 2$ at $nbrProcInst = 1500$ (i.e.

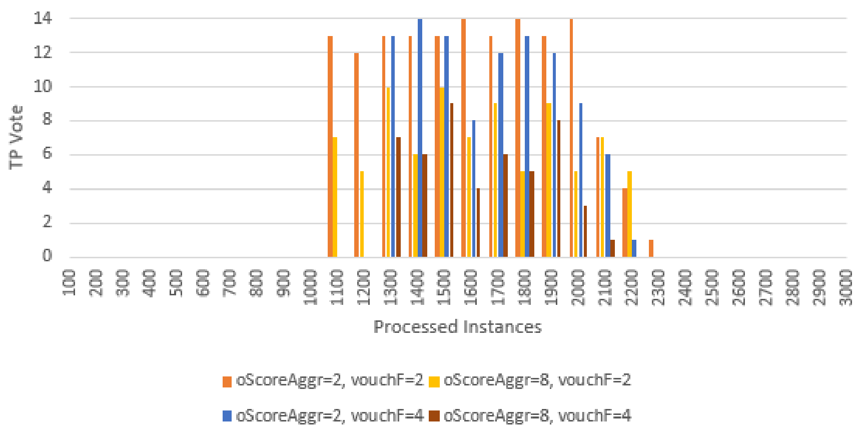


Fig. 4 TP votes generated by RandSubOut with AnyOut over $w = 100$.

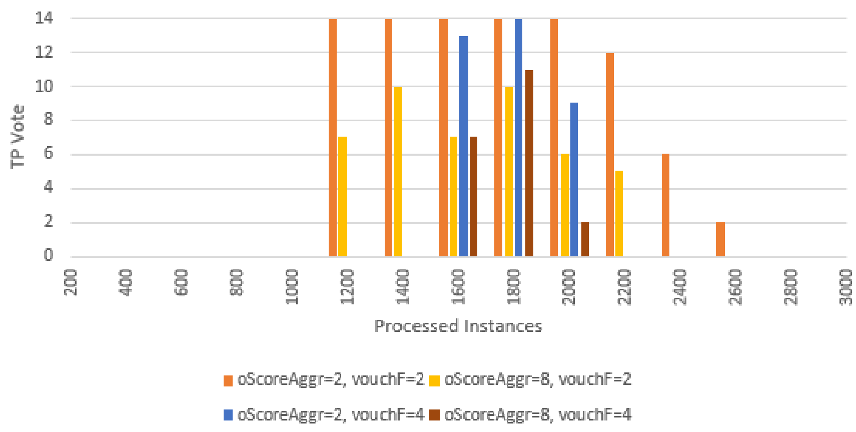


Fig. 5 TP votes generated by RandSubOut with AnyOut over $w = 200$.

$wIter = nbrProcInst \div w = 1500 \div 100 = 15$ such that $wIter$ starts at 0). Also, Fig. 2 for $k = 70, r = 0.4, w = 150$ reports a true alarm with $vote \geq 2$ for $TP = 1$ at $nbrProcInst = 1350$, where $lowerVouchBound = 1200$ and $upperVouchBound = 1500$ (i.e. $wIter = 9$). However, no case in Fig.3 satisfies the condition of a true flag alarm ONLY generated between $lowerVouchBound$ and $upperVouchBound$ with $vote \geq voteF; voteF = 2$ for $TP = 1$.

Similarly, Fig.4, and Fig. 5 show the number of TP votes generated by RandSubOut using AnyOut base learners over window size $w = 100$, and $w = 200$ respectively with varying values of parameters $oScoreAggr$ and $vouchF$. It is clear that no case in both figures satisfies the condition of a true flag alarm ONLY generated between $lowerVouchBound$ and $upperVouchBound$

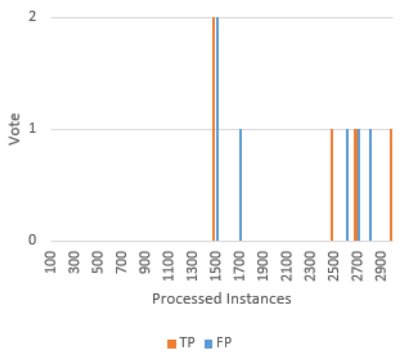


Fig. 6 TP votes vs. FP votes generated by RandSubOut with MCOD over $w = 100$.

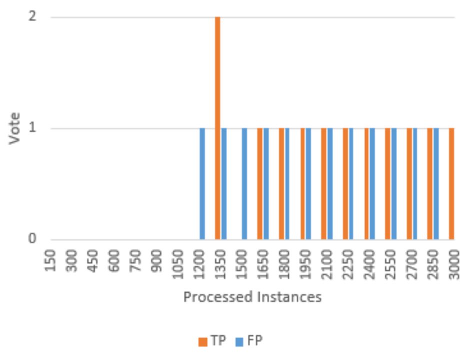


Fig. 7 TP votes vs. FP votes generated by RandSubOut with MCOD over $w = 150$.

with $vote \geq voteF; voteF = 2$. Moreover, the results revealed flag alarms with a great number of votes which do not map to the malicious insider threat scenario. For instance, the worst case of $vote = 14$ implies that 14 feature subspaces voted for this flag alarm as true, so approximately 14 or less random features reported a malicious insider threat which is not the case. Mapping to the malicious insider threat scenario considered, only few features such as *freq. of connecting a device* and *freq. of copying files to removable media device* would only report a malicious insider threat. The reason behind those unstatisfying results correlate to that an outlier score of an instance x_i is computed once a new instance x_{i+1} arrives. This means that the processing is interrupted and the confidence of the outlier score computed would be very low.

Based on the above experiments, we select two cases of RandSubOut with MCOD base learners. The first case for $k = 60, r = 0.5, w = 100$ is reported in Fig. 6 showing the TP votes vs. FP votes with respect to processed instances. The second case for $k = 70, r = 0.4, w = 150$ is reported in Fig. 7 showing the TP votes vs. FP votes with respect to processed instances. Fig. 6 flags a true alarm with $vote = 2$ for $TP = 1$ and a false alarm with $vote = 2$ for $FP = 1$ at $nbrProcInst = 1500$. However, Fig. 7 only flags a true alarm with $vote = 2$ for $TP = 1$ at $nbrProcInst = 1350$, but it didn't flag any false alarm with $vote \geq 2$. Thus, in this case $FP = 0$ where no false alarms are generated. Hence, the vote goes to RandSubOut with MCOD base learners for $k = 70, r = 0.4, w = 150, vouchF = 2, voteF = 2$.

5 Conclusion

This paper addresses the problem of high number of false alarms in the existing insider threat detection approaches from the perspective of stream mining data where a concept drift is an indication of an insider threat. We present RandSubOut approach for insider threat detection over real-time data streams. RandSubOut allows to detect insider threats represented as localised outliers over random feature subspaces, which may not be detected over the whole feature space. It also introduces two parameters: a vouch factor to confirm whether a flag alarm should be generated over a feature subspace, and a vote factor to confirm whether a flag alarm should be generated as a total vote of the ensemble. We evaluated RandSubOut as an ensemble of MCOD and AnyOut outlier detection methods, according to evaluation measures including TP and FP. The experiments revealed that RandSubOut with MCOD base learners outperforms RandSubOut with AnyOut base learners. Moreover, RandSubOut with MCOD base learners for $k = 70, r = 0.4, w = 150 vouchF = 2, voteF = 2$ provided the best results, where it votes for true alarms with $vote \geq 2$ for $TP = 1$ at real-time or near real-time. In this case, the votes for FP are ≤ 2, so no false alarms were generated.

Future work will be to address malicious insider threat scenarios that map to the different patterns of concept drift other than outliers. It will also tackle how to distinguish between a normal drift and a concept drift over real-time data streams.

References

[1] Angiulli F, Fassetti F (2010) Distance-based outlier queries in data streams: the novel task and algorithms. Data Mining and Knowledge Discovery 20(2):290–324

[2] Assent I, Kranen P, Baldauf C, Seidl T (2012) Anyout: Anytime outlier detection on streaming data. In: International Conference on Database Systems for Advanced Applications, Springer, pp 228–242

[3] Axelrad ET, Sticha PJ, Brdiczka O, Shen J (2013) A bayesian network model for predicting insider threats. In: Security and Privacy Workshops (SPW), 2013 IEEE, IEEE, pp 82–89

[4] Azaria A, Richardson A, Kraus S, Subrahmanian V (2014) Behavioral analysis of insider threat: a survey and bootstrapped prediction in imbalanced data. IEEE Transactions on Computational Social Systems 1(2):135–155

[5] Barrios RM (2013) A multi-leveled approach to intrusion detection and the insider threat. Journal of Information Security 4(01):54

[6] Bin Ahmad M, Akram A, Asif M, Ur-Rehman S (2014) Using genetic algorithm to minimize false alarms in insider threats detection of information misuse in windows environment. Mathematical Problems in Engineering 2014

[7] Chen Y, Malin B (2011) Detection of anomalous insiders in collaborative environments via relational analysis of access logs. In: Proceedings of the first ACM conference on Data and application security and privacy, ACM, pp 63–74

[8] Chen Y, Nyemba S, Malin B (2012) Detecting anomalous insiders in collaborative information systems. IEEE transactions on dependable and secure computing 9(3):332–344

[9] Gama J, Žliobait I, Bifet A, Pechenizkiy M, Bouchachia A (2014) A survey on concept drift adaptation. ACM Computing Surveys (CSUR) 46(4):44

[10] Georgiadis D, Kontaki M, Gounaris A, Papadopoulos AN, Tsichlas K, Manolopoulos Y (2013) Continuous outlier detection in data streams: an extensible framework and state-of-the-art algorithms. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, ACM, pp 1061–1064

[11] Glasser J, Lindauer B (2013) Bridging the gap: A pragmatic approach to generating insider threat data. In: Security and Privacy Workshops (SPW), 2013 IEEE, IEEE, pp 98–104

[12] Huang L, Stamp M (2011) Masquerade detection using profile hidden markov models. computers & security 30(8):732–747

[13] Kontaki M, Gounaris A, Papadopoulos AN, Tsichlas K, Manolopoulos Y (2011) Continuous monitoring of distance-based outliers over data streams. In: 2011 IEEE 27th International Conference on Data Engineering, IEEE, pp 135–146

[14] Legg PA, Moffat N, Nurse JR, Happa J, Agrafiotis I, Goldsmith M, Creese S (2013) Towards a conceptual model and reasoning structure for insider threat detection. JoWUA 4(4):20–37

[15] Nurse JR, Legg PA, Buckley O, Agrafiotis I, Wright G, Whitty M, Upton D, Goldsmith M, Creese S (2014) A critical reflection on the threat from human insiders–its nature, industry perceptions, and detection approaches. In: International Conference on Human Aspects of Information Security, Privacy, and Trust, Springer, pp 270–281

[16] Parveen P, Weger ZR, Thuraisingham B, Hamlen K, Khan L (2011) Supervised learning for insider threat detection using stream mining. In: 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence, IEEE, pp 1032–1039

[17] Parveen P, Mcdaniel N, Weger Z, Evans J, Thuraisingham B, Hamlen K, Khan L (2013) Evolving insider threat detection stream mining perspective. International Journal on Artificial Intelligence Tools 22(05):1360,013

[18] Punithavathani DS, Sujatha K, Jain JM (2015) Surveillance of anomaly and misuse in critical networks to counter insider threats using computational intelligence. Cluster Computing 18(1):435–451

[19] Sen S (2014) Using instance-weighted naive bayes for adapting concept drift in masquerade detection. International Journal of Information Security 13(6):583–590

[20] Tang K, Zhou MT, Wang WY (2009) Insider cyber threat situational awareness framwork using dynamic bayesian networks. In: Computer Science & Education, 2009. ICCSE'09. 4th International Conference on, IEEE, pp 1146–1150

[21] Verble J (2014) The nsa and edward snowden: surveillance in the 21st century. ACM SIGCAS Computers and Society 44(3):14–20

[22] Yang D, Rundensteiner EA, Ward MO (2009) Neighbor-based pattern detection for windows over streaming data. In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, ACM, pp 529–540

[23] Young WT, Goldberg HG, Memory A, Sartain JF, Senator TE (2013) Use of domain knowledge to detect insider threats in computer activities. In: Security and Privacy Workshops (SPW), 2013 IEEE, IEEE, pp 60–67

[24] Yu Y, Graham JH (2006) Anomaly instruction detection of masqueraders and threat evaluation using fuzzy logic. In: 2006 IEEE International Conference on Systems, Man and Cybernetics, IEEE, vol 3, pp 2309–2314

[25] Zhang R, Chen X, Shi J, Xu F, Pu Y (2014) Detecting insider threat based on document access behavior analysis. In: Asia-Pacific Web Conference, Springer, pp 376–387